البرمجة بلغة البايثون

رحلة ممتعة من الولا حاجة حتى الإحتراف



PINK PYTHON

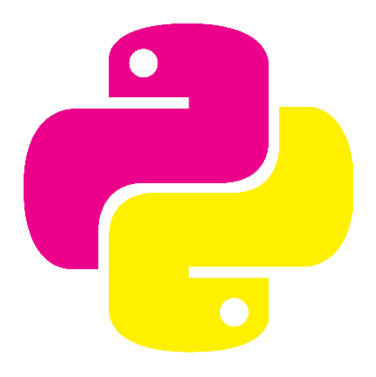
Pink PYTHON

تأليف :

م محمود على عبداللاه


بسم الله الرحمن الرحيم

نسخة الكتاب



PINK PYTHON 1.0

قد لا تكون هذه النسخة الأخيرة من الكتاب وصدر بعدها نسخ

أحدث بها تعديلات لأخطاء أو إضافة لمحتوى جديد.

فتابع هنا لمعرفة أخر إصدار releases

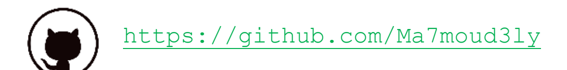
عن المؤلف

خريج كلية الهندسة قسم الإتصالات والإلكترونيات جامعة سوهاج

للتواصل/ الاقتراحات/الاستفسارات







محمود على

6 أكتوبر 2018

إهـــداء

لـكـل مـبرمـج .. مـكـتـئـب.. بائس.. حـزيـن.

والـلـى مـزنـوق فى bug بـقـالـه يـومـين

والـلـى مـطـبـق عـلـى بـروجـكـت والـنـاس نـايــمـين

والـلـى عـايش عـلـى الـقـهـوة والـكـافـيـين

والـفـرى لانـسـرز الـشـقـيـانـين...

والـلـى كـل أكـواده مـن stackoverflow

والـلـى الـبـويـنـتـرز والـسـيـمـى كـولـون جـنـنـوه

وابـنـنـا الـهـاكـر الـلـى الـ CIA هـيـغـتـالـوه...

والـلـى أنـتـحـر من طـلبات عـمـيـل

والـلـى بـيـقـول عـلـى أكـونـت الـفـيـس إيــمـيـل..

والـلـى عـايـز يـبرمـج أنـدرويـد عـلـى بـانـتـيـوم 4

والـلـى بـيـخـلـى رقـم تـلـفـونه كـلـمـة مـرور...

والـلـى بـيـدور عـلـى كـراك بـرنـامـج ومـش عـايـز يـشـتـريـه

والـلـى سـجـل فى ألـف كـورس ونـام عـلـيـه

والـلـى مـالـهـوش فـيـهـا.. ومـش فـاهـمـنـي بـقـول إيـه..

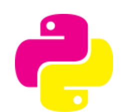
Pink Python..

intentionally blank page

هذه الصفحة تركت فارغة عمداً



فصول الكتاب



مقدمة عن الكتاب

فى البداية كده وقبل ما نبدأ عاوزين نعرف شوية حاجات.

يعنى أيه بايثون 🤔

لو كنت حملت الكتاب بالصدفه او جذبك الكوفر أو مش عارف يعنى ايه بايثون..

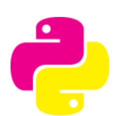
فـ ببساطه شديده بايثون دى لغة برمجة , ولغة البرمجة Programming language عبارة عن مجموعة من القواعد والتعليمات اللى بنوجهها للكمبيوتر عشان ينفذها فى شكل البرامج والتطبيقات والمواقع والألعاب اللى بنشوفها وبنتعامل معاها.

يعنى لغة البرمجة دى حاجة بتعمل بيها برامج وتطبيقات,ولغة البايثون اللى بنتكلم عنها فى الكتاب تعتبر واحدة من أسهل وأشهر لغات البرمجة فى العالم.

مين يقرى الكتاب ده 🤔

لو كنت متعرفش أى حاجة نهائى عن البرمجة.. أو كنت متخصص و مهتم بالمجال بسبب الداراسة أو بتتعلم مع نفسك.. لو كنت عاوز تتعلم برمجة ومش عارف تبدأ بأى لغة.. لو كنت حابب تعمل تطبيقات للموبايل والكمبيوتر وفاكر إن الموضوع صعب أو مستحيل.. لو عاوز تتعلم لغة برمجة سهلة و مطلوبة فى سوق العمل.

أياً كان سنك وخبرتك فى الكمبيوتر هتلاقى كتاب Pink Python نازل لمستواك بشرح وأسلوب مبسط وطالع معاك لمستوى متقدم بداية من أساسيات اللغة مروراً بمواضيع قواعد البيانات واستخدام مكتبات الذكاء الإصطناعى و معالجة الصور وتحرير الفيديوهات وبرمجة المواقع وتطبيقات والموبايل وكله ده بلغة البايثون.



هحتاج إيه عشان أشتغل معاكم 🤔

مش لازم يكون عندك كمبيوتر او لابتوب بإمكانيات خارقة عشان تتعلم لغة البايثون.
ومش لازم يكون عندك كمبيوتر أصلاً تقدر تنزل مفسر اللغة على الموبايل وتكتب أكواد و تتعلم معانا الأساسيات من غير أى مشاكل.او تقدر تشتغل من المتصفح أونلاين فملكش أى حجة.

ملاحظات عن أسلوب الكتاب

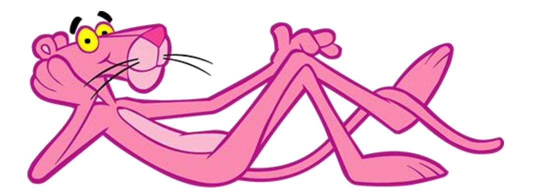
- فضلت إن الكتاب يكون بالعامية المصرية وده لسببين : أحدهما إنها بتوصل المعلومة بشكل أوضح و أبسط والثانى إن مؤلف الكتاب مش الأستاذ انيس منصور اللّه يرحمه… وكون موضوع الكتاب علمى او برمجى ده ممنعنيش إطلاقاً من استخدام إيموشناتى المفضلة 😁.
- بعض المصطلحات البرمجية ليها ترجمات مش مرضيه بالنسبه لى فهتلاقينى بجيب الكلمة بالإنجليزى وجنبها الترجمة بالعربى.. والمصطلحات الغريبة بضربلها ترجمة واقعية و مفهومة من عندى 😁.
- وجود أى أخطاء إملائية/لغوية/برمجية/لينكات مش صحيحة/تنسيق مش مزبوط. شىء وارد جداً وهكون مبسوط لو بلغتنى بيها أو بأى أستفسار برمجى على صفحة الكتاب على الـ Github issues أو على أحد اكونتاتى الشخصية.

أخيراً أتمنى أن يحقق الكتاب هدفه الرئيسى بأن يكون كتاب مجانى مبسط ممتع و شامل ومناسب لأى حد عنده شغف أو أهتمام بالبرمجة, ولا اسألكم عليه أى حاجة… بس وجهوا دعواتكم بأنكم توصلوا لآخر الكتاب على خير.

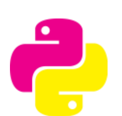
(وَرَبُّنَا الرَّحْمَنُ الْمُسْتَعَانُ عَلَى مَا تَصِفُونَ)



الفصل 0 – ليه بايثون ؟



الفهرس



لغة البايثون Python

من التعريف الموجود على الويكيبيدا.. لغة البايثون Python

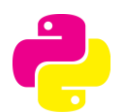
Python is an interpreted high-level programming language
for general-purpose programming.

طبعاً مش عايز أعملك ترجمة للتعريف تصعب عليك فهمه أكتر وهنرجع نتكلم عنه نقطه نقطة, بس أعرف شوية عن تاريخ اللغة والهدف منه وهنرجعله..

لغة البايثون أبتكرها المبرمج الهولندى Guido Van Rossum سنة 1989.. كان فاضى فى أجازة الكريسماس فحب يعمل حاجة تشغله فقال يعمل لغة برمجة language scripting جديدة, وبدأ يشتغل فى لغة البايثون واللى تأثر جداً في كتابتها بلغة برمجة اسمها ABC .. وسنة 1999 الأخ فان روسوم عمل بروبوزال proposal بعنوان Computer Programming for Everybody بيشرح فيها أهدافه من لغة البايثون واللى تحققت بشكل كبير ودلوقتى أقدر أعرضهالك كمميزات للغة البايثون.

مـميزات لغة البايثون

- لغة سهلة وبسيطة ولا تقل قوة عن اللغات المنافسة.
- لغة مفتوحة المصدر أى حد يقدر يساعد فى تطويرها
  ونتيجة ذلك بقى ليها مجتمع كبير.
- لغة قريبة من لغة الإنسان و أكوادها سهلة الفهم.
- لغة مناسبة لجميع الأغراض والاستخدامات البرمجية.

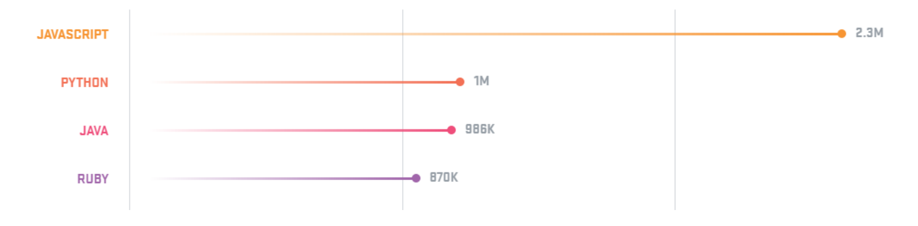


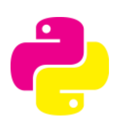
فنتج عندنا لغة سهلة التعلم easy-to-learn وسهلة الفهم easy-to-read و portable بمعنى إن نفس الكود أو مع إضافات بسيطة تقدر تشغله على جميع أنظمة التشغيل Windows, Linux, Mac وحتى على أنظمة تشغيل الموبايل android, ios وسيرفرات الإنترنت.. وده سبب أنتشار اللغة بشكل كبير وزيادة شعبيتها.

فى الوقت الحالى لغة البايثون كانت بتحتل المركز التالت فى تصنيف موقع Tiobe Index للغات الأكثر شعبية فى العالم.

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	^	Python	7.653%	+4.67%
4	3	v	C++	7.394%	+1.83%
5	8	^	Visual Basic .NET	5.308%	+3.33%

ده غير المركز الثانى بعد لغة الجافاسكريبت فى اللغات الأكثر استخداماً على موقع GitHub .



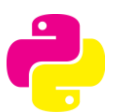


عودة للتعريف..

Python is an interpreted high-level programming language
for general-purpose programming.

أول حاجة لغة البرمجة programming language : هى مجموعة من الأوامر بنتعامل بيها مع أجهزة الكمبيوتر ومرت بعدة مراحل فى تطورها بداية من التعامل مع معالج الكمبيوتر مباشرة لحد ظهور لغات البرمجة عالية المستوى المعروفة واللى بتختلف عن بعضها فقط القواعد والأساليب البرمجية ولكن أغلب اللغات ليها نفس المبدأ والأساسيات.

- فى البداية لازم نعرف إن أجهزة الكمبيوتر أو بشكل ادق المعالجات processors مش بتفهم غير لغة الآلهmachine code اللى هى عبارة عن أكواد مكونة من أصفار ووحايد zeroes ones بتتحول لإشارات كهربية بيتعامل معاها البروسيسور.
- ولأن موضوع التعامل بلغة الآله صعب على الإنسان قرروا يسهلوا الموضوع شوية فعملوا لغة برمجة تتعامل بشكل مباشر مع الرامة والبروسيسور برده ولكن بأوامر يقدر يفهمها الإنسان زى MOV يعنى انقل ADD أجمع.
- يعنى اللغة تكلم البروسيسور وتقول له روح خد القيمة الموجودة فى مكان location معين فى الرامة وخزنها فى الـ register الفلانى (مكان تخزين فى البروسيسور) وهات عليها قيمة تانية وأجمعها وحط الناتج فى مكان فى البروسيسور أو طلعه على بورت معين... اللغة دة أسمها لغة الأسمبلى Assembly ولأنها بتتعامل بشكل مباشر مع البروسيسور والرامة ووحدات الأدخال/الإخراج فبيسموها Low level language .



- وبعدين حاولوا يبسطوا الموضوع أكتر وقالوا عاوزين نعمل لغات تبعد المستخدم عن التعامل مع الهاردوير بشكل مباشر, وتكون سهلة الفهم والقراءة..ومتعتمدش على نوع البروسيسور اللى هيشغل البرنامج ده..لأنك لما تبرمج بلغة الأسمبلى لازم تعرف البنية بتاعة المعالج عشان بتتعامل مع الحاجات اللى جواه وكل معالج مع التطور حجم ريجستراته ونواقل الداتا فيه بيزيد أو بيتغير.
- فقرروا يعملوا لغات أكثر سهولة وأقرب للغة الإنسان وسموها لغات عالية المستوى أو High Level Language وطبعاً كل اللى اللغات المشهورة واللى بنستخدمها فى الوقت الحالى واللى منها بايثون من النوعية دة .

ده معناه إن البايثون لغة برمجة يعنى حاجة بنخلى الكمبيوتر ينفذ تاسكات من خلالها. ولغة High Level يعنى بنكتب كود سهل مفهوم لينا بلغة قريبة من لغتنا وبنشغله على أى جهاز مهما كان نوع المعالج بتاعه او حجم الرامة وهم الأتنين يولعوا مع بعض مع بقية الهاردوير ده شيء ميهمناش المهم البرنامج يشتغل😂.

لغة : Interpreted يعنى لغة مفسرة, وده هياخدنا لشوية رغى تانى..

نسيت أقولك لما تكتب الكود بلغة الأسمبلى فيه برنامج اسمه الـ assembler ده بياخد الكود ويحوله لصورة binary أو hexadecimal جميعة كده للبينارى او الأصفار والوحايد وهى دى اللغة اللى بيفهما بسلامته البروسيسور.

- فى لغاتنا الـ High Level تعالى نسمى الكود الجميل المفهوم اللى شبه لغة الانسان باسم source code .. و ده كود أنت كمرمج تفهمه أما الكمبيوتر ميفهموش.



- فلزم الأمر حاجة زى الـ assembler يحول الكود ده للغة الآله.. فعملوا حاجة اسمها complier والكومبايلر عموماً ده برنامج بيحول الداتا من صورة لأخرى, بس فى موضوعونا الـ compilers بتحول الـ source code بتاع اللغة اللى أحنا بنفهمها إلى machine code اللى بيفهمه مين .. البروسيسور.
- واللغات اللى بتدعم كده اسمها compiled language زى C/C++/BASIC وغيرهم.
- بس لقيوا مشكلة, أنا شغال على ويندوز كتبت كود بلغة الـ C وعملت له compile وحولته لملف تنفيذى .exe executable code أو binary وشغلته وتمام.

لكن لو أخدت البرنامج ده وحاولت أشغله على نظام تانى زى لينكس Linux على نفس الكمبيوتر ونفس المعالج.. البرنامج مش هيرضى يشتغل.

- والسبب إن البرنامج لما عملتله compile على نظام تشغيل معين الملف التنفيذى اللى طلع مش كله عبارة عن الـ machine code للبرنامج ..لكنه فيه جزء أو header أضافه ليه الكومبايلر ينظم طريقة تشغيل البرنامج ده لما تدوس عليه و هيتحمل ازاى ويتحط فى الرامة عشان المعالج يتعامل معاه.
- وطبعاً الأسلوب المتبع مع ويندوز يتخلف عن لينكس وبالتالى الـ executable files/binaries بتاعة Linux مش هتتشغل على windows والعكس .
- دى بقى مشكلة فى اللغات الـ compiled إنها مرتبطة بنظام التشغيل platform .dependent
- حاولنا نــحل المشكلة دى فقولنا البرنامج اللى هنكتبه بأى لغة البرمجة source code مش هنحوله للغة الآله..هنسيبه زى ماهو بصورته النصية وهنعمل برنامج نثبته

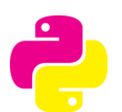


علـى كل نظام تشـغيل windows/linux/android/ios ولما أجـى أشـغل الكـود ده البرنامج هياخده وهيحوله للغة الآله وقت التشـغيل وبالتالى أقدر اكتب كـود واحد وأشـغله علـى أى نظام تشـغيل متثبت عليه البرنامج اللى هيفسره.

- البرنامج الوسيط ده اسمه interpreter وده برنامج بيحـول الكـود النصـى واللى بسميه script للغة الآله وقت التشـغيل,وكل مره أشـغل السـكربت الـ interpreter بيبدأ من أول سطر في الكـود ويحوله للغة الآله علـى عكـس الـ compiler اللـى بيحـول الكـود إلى machine code مرة وحده.

الكلام ده عشان تفهم معنى إن لغة scripting language يعنى البرنامج اللى بعمله بيفطل طول عمره بصورته النصيه وأى حد يقدر يشوف الكود.. ولغة interpreted يعنى السكربت بيتحول للغة الآله وقت التشغيل.

وأخر نقطة فى التعريف .. لغة general purpose يعنى مش مخصصة لمجال معين وتقدر تستخدمها فى أى حاجة والجزئية الجاية هنعرف أنواع البرامج والتطبيقات اللى ممكن تعملها بلغة البايثون.



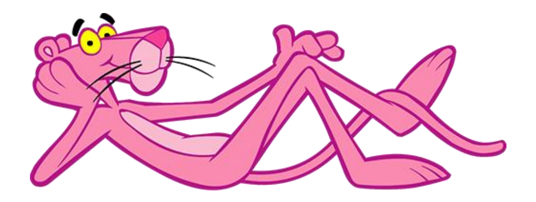
استخدامات لغة البايثون

- تطبيقات سطح المكتب desktop application
- تطبيقات الويب والمواقع web application
- تطبيقات الموبايل android/ios
- جمع وسرقة المحتوى من المواقع web scrapping
- فى التطبيقات العلمية والحسابية بديل مجانى للغة الماتلاب
- تطبيقات الذكاء الإصطناعى AI وتعلم الآله Machine Learning.
- برمجة الشكبات Networking وإدارة السيرفرات.
- فى الأنظمة المدمجة و الربوت وبرمجة راسبيرى باى raspberrypi
- فى تعليم لغات البرمجة بأعتبارها لغة ساهلة وتصلح للاطفال والمبتدئين

وعشان تعرف أكتر عن التطبيقات اللى تقدر تعملها بلغة البايثون تقدر تشوف الموضوع ده على الموقع الرسمى من apps.



الفصل 1 – البرمجة بلغة بايثون



تثبيت لغة البايثون

- ✓ على ويندوز Windows
- ✓ على أنظمة لينكس Linux
- ✓ على الموبايل Android و IOS
- ✓ بايثون اونلاين Python Online
- ✓ أساليب البرمجة بلغة البايثون

الفهرس



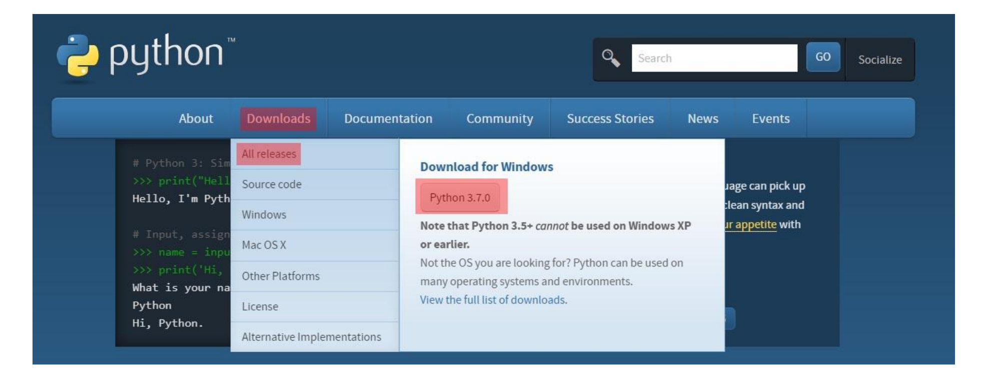
تثبيت لغة البايثون

عرفنا فى الفصل السابق إن لغة البايثون من اللغات المفسرة Interpreted وبنقدر نشغل الكود اللى بنكتبه بلغة البايثون واللى سمناه سكريت script على أى نظام تشغيل عليه مفسرة لغة البايثون..يعنى دلوقتى أياً كان نظام التشغيل بتاعك هتحتاج يكون على جهازك بيئة تطوير لغة البايثون python environment ودى عبارة عن الـ interpreter و محرر أكواد بتكتب فيه السكريتات بالإضافة إلى المكتبات الأساسية اللى بتيجى مع اللغة واللى هتستخدمها فى التطبيقات المختلفة.

فى الفقرات الجاية هنعرف طرق تثبيت لغة البايثون على أنظمة التشغيل والأجهزة المختلفة.

نظام تشغيل ويندوز Windows

من الموقع الرسمى للغة البايثون python هتحمل الإصدار المناسب ليك .. فى الكتاب ده هنشتغل على بايثون 3 وبالتالى تقدر تحمل أخر release متوفرة.





الفرق بين بايثون 3.x و بايثون 2.x

جايز تكون دقيق الملاحظة وانت وبتدور فى releases أو الإصدارات المختلفة للغة ملقتش الإصدارات طالعة بالترتيب او إنهم لازالوا بيعملوا تحديثات للإصدارات 2.X بالإضافة إلى 3.X وإيه الفرق بين الإصدارات الكتير دى وانا المفروض أتعلم مين فيهم ؟

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes
Python 3.6.5	2018-03-28	Download	Release Notes
Python 3.4.8	2018-02-05	Download	Release Notes
Python 3.5.5	2018-02-05	Download	Release Notes

View older releases

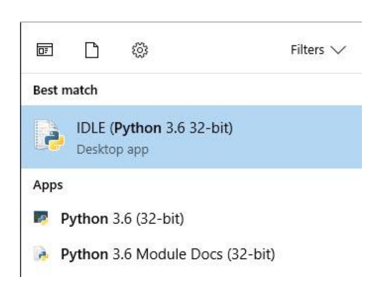
أيه الفرق؟ مالناش دعوة لأنك لحد دلوقتى متعرفش حاجة عن اللغة.. طيب هنشتغل على بايثون 2 ولا 3 ؟! الجواب أخر إصدار نازل على الموقع من بايثون 3.

على الموقع الرسمى which-python بيقول إن بايثون 2.7 الأكثر استخداماً فى المشاريع ولكنها هيتوقف الدعم والتحديثات الأمنية لها فى 2020 يعنى خلو مشاريعكم متوافقة مع على بايثون 3 بكرامتكم 😂.

لو الفضول قتلك وعاوز تعرف الفرق بين بايثون 2 و 3 والمفروض تتعلم مين فيهم فتقدر تشوف المقالة دى Python2orPython3 واللى أختصروا فيها الفرق بين الإصدارين..

Short version: Python 2.x is legacy, Python 3.x is the present and future of the language





بأفتراض إنى أقنعتك إنك تحمل أى إصدار بايثون 3 ولا سيما كان الإصدار الأخير و ثبته على جهازك هتروح تبحث فى قائمة ابدأ تكتب كلمة python أو IDLE

لو هتشتغل على أى لغة برمجة هتسمع عن حاجة اسمها IDE الكلمة دى معناها ايه ؟

Integrated development environment

An integrated development environment (IDE) An IDE normally consists of a source code editor, build automation tools, and a debugger. Most modern IDEs have intelligent code completion. Some IDEs, such as NetBeans and Eclipse

بيقولك إن الـ IDE يعنى بيئة التطوير المتكاملة, وهو عبارة عن برنامج فيه محرر نصوص وده مكان بتكتب في الكود بتاعك بأى لغة كانت , وفيه أداوت بتحول الكود ده للغة الآلة وتشغله وتطلع الاخطاء فى الكود إن وجدت..

- بالنسبة للغة البايثون على نظام تشغيل ويندوز بيجى معاها برنامج اسمه IDLE وده أختصارة الجملة integrated development and learning environment.
- المهم هتفتح برنامج Python IDLE وهتشوف الشكل اللى قدامك وهو مفسر لغة البايثون الـ Interpreter.. وبكده نكون نكون ثبتنا لغة البايثون على ويندوز وجاهزين نكتب ول كود لينا.



Python 3.6.3 Shell

File Edit Shell Debug Options Window Help

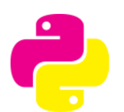
```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

بس قبل ما نبدأ مش عايزين ننسى اخواتنا الغلابه بتوع لينكس أو البؤساء اللى معندهمش كمبيوتر وعاوزين يشتغلوا من على الموبايل..

أنظمة لينكس Linux

الحقيقة مستخدمين توزيعات لينكس مش غلابة ولا حاجة وعارفين إنهم مش محتاجين ينزلوا أى برامج يا دوب يفتحوا التيرمنال terminal ويكتبوا فيه python أو python3 حسب إصدار لغة البايثون المتثبت على التوزيعة اللى شغالين عليها وهيشتغل مفسر لغة البايثون على طول.

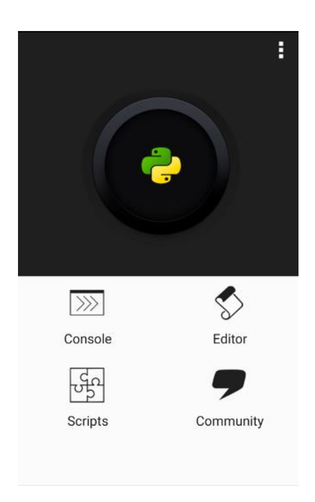
```
Python 2.6.6 (r266:84292, Aug 19 2016, 11:10:19)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



على الموبايل Android و IOS

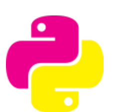
من الجميل إنك تقدر تكتب أكواد وسكريتات للغة البايثون على السمارت فوت بتاعك من غير ما يكون عندك كمبيوتر.. ومش بس كده van Rossum بيضحي النهارده 😁 كمان تقدر تعمل سكريتات لتطبيقات أندرويد كاملة من على الموبايل..وده بفضل أبليكيشن qpython3 أو qpython اللي تقدر تحمله من على جوجال بلاي أو الآب ستور وطبعاً مش محتاج أفهمك إن qpython3 ده للغة بايثون 3.

لما تفتح التطبيق على الموبايل هيكون شكله كده ..





و تقدر من خلال الزرار Editor تفتح محرر اللغة وتكتب سكريتات بلغة البايثون أو من الـ Console تفتح مفسر اللغة التفاعلي الـ Interactive Interpreter اللي هنشوف بعد شوية تفاعلي ازاي وهنعمل بيه أيه 😁.



بايثون اونلاين Python Online

- تقدر تجرب اكواد لغة البايثون اونلاين من غير ما تحتاج أنك تحمل أى برامج, وطبعاً فيه مواقع كتير تقدر تجرب عليها فلو مش عاوز تسجل وعاوز تجرب على السريع فبرشحلك execute_python_online على موقع tutorialspoint.
- او ممكن تستخدم موقع PythonAnyWhere وده من المواقع الجميلة اللى هنستخدمها فى نشر تطبيقات الويب اللى هنعملها بالبايثون فى الفصل قبل الاخير لو كان لينا عمر 😁.. وتقدر تسجل حساب مجانى على الموقع ده اللى بيديك أمكانية أختيار أصدار البايثون المناسب ليك وتقدر تثبت المكتبات اللى هتحتاجها عليه.

بعد ما تسجل دخول على الموقع هتفتح صفحة Dashboard وهتختار مفسر البايثون بالإصدار المناسب..

pythonanywhere Dashboard Consoles Files Web Tasks Databases

Dashboard

CPU Usage: 0% used – 0.00s of 100s. Resets in 15 hours More Info
File storage: 21% full – 105.2 MB of your 512.0 MB quota

Upgrade Account

Recent
Consoles

You have no recent consoles.

View all

New console:
$ Bash >>> Python More...
Version
3.7 3.6 2.7

Recent
Notebooks

Your account does not support Jupyter Notebooks. Upgrade your account to get access!

All
Web apps

Open Web tab



أسـاليـب البرمجـة بلـغة البـايـثون

تعالـو نـكتـب اول كـود لـينا بـلغـة البـايـثـون, وهـنـعـرف إن فيـه 3 طـرق عـشـان تـكـتـب وتـشـغـل الاكـواد بتاعتك.

النظـام التـفاعـلي Interactive Mode

فى لـغة البـايـثون ومـعـظـم اللـغات المفسـرة فيـها نظام البرمجـة بشـكـل بتفـاعـلـي.. بإنـك تـكـتـب الكـود البرمجـي سـطـر سـطـر فى الـ Interpreter كـأنك بتعـمـل chat مع الكـمبيـوتر.

- عشـان تشـتـغـل معـانا مع المفسـر التـفاعـلي.. لـو عـلـى ويـنـدوز من قـائمـة أبـدأ هـتـفـتـح برنامج الـ IDLE أو هـتـفـتـح مـوجـه الأوامر CMD وتـكـتـب فيـه python.
- ولـو عـلـى المـوبايل هـتـفـتـح تطبيـق qpython وتختار console
- ولـو عـلـى أى تـوزيـعـة لـنـظـام لـيـنـكـس هـتـفـتـح الـ terminal وتـكـتـب فيـه python.

لما يـفـتـح الـ Interactive Interpreter هـتـكـتـب فيـه الـكـود ده وتـدوس Enter

```
print("Hello World ^_^")
```

شـكـل الـ IDLE عـلـى ويـنـدوز

```
Python 3.6.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/Mahmoud/AppData/Local/Programs/Python/Python36-32/testsss.py
>>> print("Hello World ^_^")
Hello World ^_^
>>> |
```



شكل المفسر فى الـ CMD على ويندوز ومتستناش تشوفه على ليونكس عشان مش هرسترت الجهاز 😁.

```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Mahmoud>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World ^_^")
Hello World ^_^
>>>
```

شكل المفسر على موقع PythonAnyWhere

```
Python3.7 console 10543633                                Share with others
Python 3.7.0 (default, Aug 12 2018, 20:40:44)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World ^_^")
Hello World ^_^
>>>
```

- المهم فى الكود ده أنت استخدمت الدالة print عشان تطبع الجملة المشهورة اللى بتعملها ولما تبدأ فى أى لغة برمجة Hello World وظهرت قدامنا على الشاشة.

ولو مستقل باللى أنت عملته فشوف الكود اللى بيطبع نفس الجملة بلغة Java ولغة C++ وأعرف العز اللى انت فيه 😁.

```
public class Hello
{
        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }
}
```

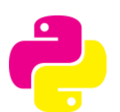
Java

```
#include <iostream>

using namespace std;

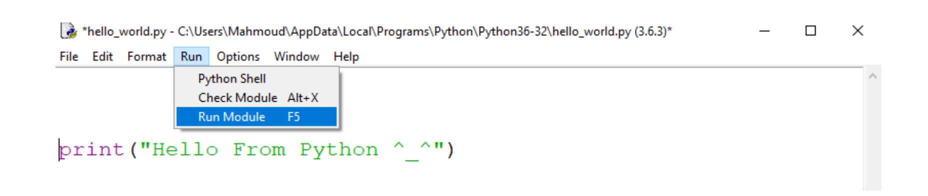
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

C++



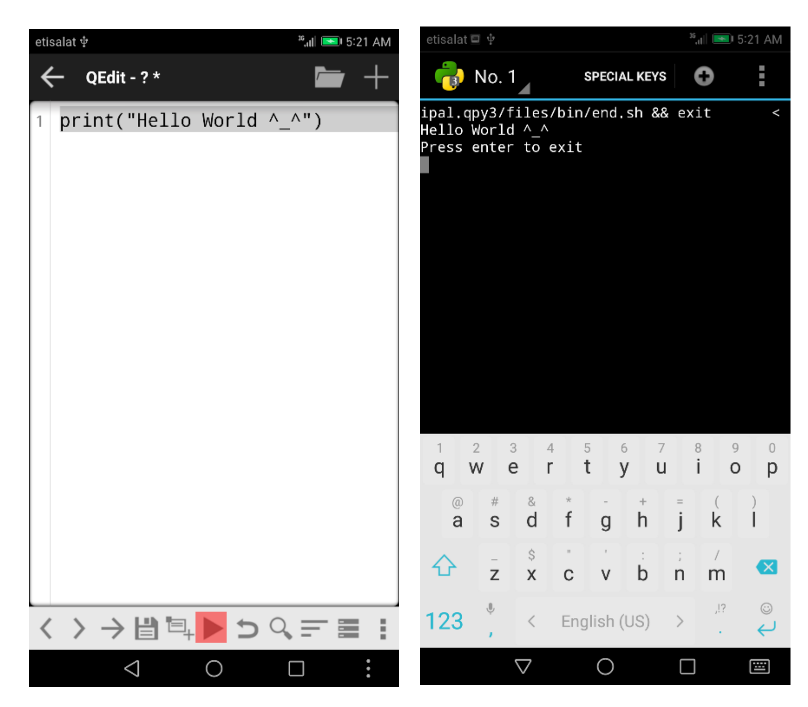
كتابة سكريت على Python IDLE

- لو شغال على ويندوز وفتحت الـ IDLE من قائمة File > New File أو تأكد إن لوحة المفاتيح إنجليزي ودوس Ctrl+N وهيفتحلك محرر اكواد لغة البايثون..وهو code editor متواضع جداً مكتوب بلغة البايثون.
- وهنسخ فيه الكود ده print("Hello World ^_^") وتشغل السكريت عن طريق إنك تدوس F5 او من قائمة Run > Run module
- لو مكنتش حفظت السكريت هيطلب منك تحفظه وبعدها هيفتح السكريت وينفذه في الـ Interactive interpreter



- وعشان تكتب سكريت بايثون على برنامج Qpython على الموبايل.. هتشغل البرنامج وتفتح الـ Editor وتعمل سكريت جديد تكتب فيه الكود وتحفظه بأي اسم مثلاً test.py وهيشتغل معاك.





محررات الأكواد Code Editors

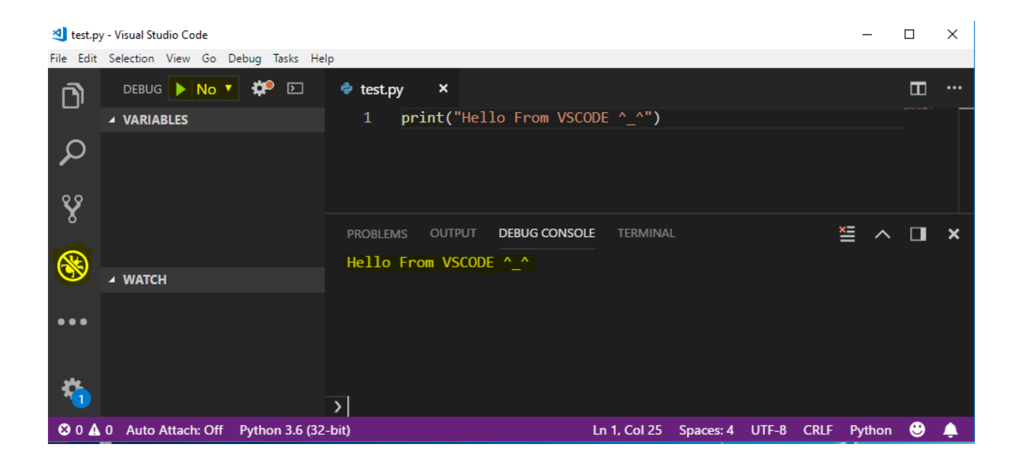
لو مش حابب برنامج Python IDLE تقدر تستخدم أي code editor مشهور زي sublime notepdad++ – vscode بتكتب عليه السكريبت بتاعك وتحفظه وتعمله تشغيل من خلال الـ Terminal في لينكس أو الـ CMD على ويندوز.

وبرشحلك برنامج فيجوال ستوديو كود vscode برنامج تبع مايكروسوفت مجاني ومفتوح المصدر وبتقدر تضيف عليه extensions للغة البايثون.. وفيه ميزات كتير عن أي code editor تاني في سهولة تنزيل الـ extensions وتكملة الأكواد وكشف الاخطاء وغيرها..

تقدر تحمل البرنامج لنظام تشغيلك من هنا visualstudio



- كل اللى هتعمله إنك تفتح ملف جديد وتحفظه بصيغة py. مثلاً test.py وعلى طول هيظهر hint يطلبك منك تثبت extension للغة البايثون و بعد ما تثبتها تعمل reload للبرنامج عشان يفعل الـ extension الجديدة.
- هتكتب الأكواد بتاعتك ومن قائمة debug على الشمال هتدوس على start debugging وهيشتغل السكريبت وهتشوف الناتج بتاعه فى الـ DEBUG CONSOLE من تحت.



طبعاً فيجوال ستوديو كود فيه debugger بيخلينا نقدر نشغل الـ script من البرنامج نفسه شبه الـ IDLE .. لكن لو شغال على أى محرر تانى مفيهوش debugger زى notepad الحقيرة بتاعة ويندوز هتحتاج تشغل الكود بنفسك فى الـ command line.



تشغيل السكريتات فى الـ command Line

الـ command line أو سطر الأوامر وهو الشاشه السوده اللي اسمها CMD ومبتعملش بيها أى حاجة على ويندوز أو الـ Terminal اللي بتستخدمه على لينكس وبتتخيل نفسك هاكر😁.. المهم فى الحالة دى هتكتب الكود بتاعك بأى برنامج محرر نصوص وبتحفظه بامتداد py. وبتشغله فالـ command line بالطريقة دى.

```
python script_name.py
```

- مثلاً فى أى مكان هتعمل ملف جديد وتسميه test.py وتكتب فيه الجملة اللى بنطبعها من بدرى وتفتح الـ Command line توجهه للمسار اللى موجود فيه السكريت ده.

```
test.py - Notepad
File Edit Format View Help
print("Hello From Command Line ^_^")
```

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Mahmoud>E:

E:\>python test.py
Hello From Command Line ^_^
```

- هنا انا حافظ السكريت test.py فى البارتيشن E عشان متقولش إيه اللى وداك هنا.

و طبعاً لازم تكون بتعرف تتعامل مع الـ command line عشان تستخدم الأخيرة.



وأخيراً واحد هيسألنى هنستخدم انهى طريقة أو برنامج 🤔؟!

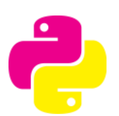
هتعدى عليك مراحل هتحتاج تستخدمهم كلهم.. بس فى البداية وأنت وبتتعلم هتشتغل على الـ Interactive Interpreter على الـ IDLE أو الـ command line لأنه هيساعدك تشوف كل سطر وجملة فى البرنامج بتتنفذ ازاى .. وبعد كده تبدأ تكتب سكربتات على الـ Code Editor بتاع الـ IDLE أو لما تزهق منه هتختار أى Code editor او IDE تانى تشتغل عليه.. وبالمناسبة وقبل ما أخلص الشابتر ده هستغل باقى الصفحة دى وأعرفك على برنامج أو IDE عظيم اسمه PyCharm تقدر تكتب عليه سكربتات بلغة البايثون وتستخدمه فى المشاريع الكبيرة.

PC PyCharm

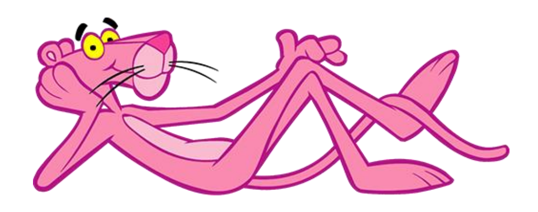
Python IDE
for Professional Developers

DOWNLOAD NOW

Full-fledged Professional or Free Community



الفصل 2 – أساسيات لغة البايثون Python Basics



الفهرس



العمليات Operations

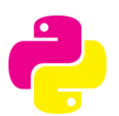
العمليات الحسابية Arithmetic Operators

دلوقتى هنتعامل مع المفسر التفاعلى بتاع لغة البايثون كأنه آله حاسبة هنخليه ينفذ العمليات الحسابيه زى الجمع الطرح والقسمة والضرب والأس وباقى القسمة.

- هفتح الـ Interactive Interpreter وهتنفذ العمليات دى..

+ للجمع	`>>> 5 + 5` `10`
* الضرب	`>>> 5 * 5` `25`
- الطرح	`>>> 5 - 5` `0`
/ القسمة	`>>> 10 / 5` `2.0`
% باقى القسمة	`>>> 5 % 2` `1`
** الأس	`>>> 5 ** 2` `25`

- العلامات دى * + - / % ** أسمها معاملات Operators.. والأرقام اللى بيعمل عليها العملية الحسابية اسمها Operands.



- تقدر تنفذ أكتير من عملية حسابية مرة وحده بس ده هياخدنا لموضوع تانى وهو أولوية العمليات الحسابية ..يعنى مثلاً لما يبقى فيه جمع وضرب وطرح هينفذ مين الاول؟
- العمليات الحسابية يتم تنفيذها بالالولية دى

  الأس << الضرب/القسمة/باقى القسمة<<الطرح/الجمع
- مثلاً العملية الحسابية دى ليه نتيجتها 1 ونفذها ازاى اصلا ؟

```
>>> 10-2*8+7
1
```

- الأولوية فى العمليات الحسابية للضرب وبعدين الجمع والطرح

  فهتبقى 7+8*2-10 = 7+16-10
- طيب لو عاوز الطرح يتنفذ قبل الضرب فيه حل ؟

```
>>> (10-2)*8+7
71
```

ايوه فيه حاجة رتبتها أعلى من الضرب وهى الأقواس ( )..

أى عملية حسابية عايزها تتنفذ الاول حطها بين أقواس دائرية..

فهنا نفذ ما بداخل القوس الأول 10-2 فالعلمية بقيت 7+8*8 بعدين الضرب 7+8*8

= 64+7

- طيب لو فيه معامل الأس ** هيتنفذ قبل الضرب

```
>>> 3+2*4**2-30/2
20.0
```

والضرب والقسم...3+2*4**2-30/2

= 3+2*16-30/2 بعدين الضرب والقسمة 3+2*16-30/2 = 3+32-15..

يبقى خليك فاكر ترتيب العمليات الحسابية فى لغة البايثون

ما بين الأقواس << الأس << الضرب / والقسمة / باقى القسمة << الجمع والطرح..



عمليات المقارنة Comparison (Relational) Operators

لو عاوز تعمل مقارنة بين قيمة وقيمة تانية بتستخدم معاملات المقارنة Relational Operators .

معاملات المقارنة بتقارن بين معاملين Operands يمين وشمال العلامة وبترجع True في حالة صحة المقارنة وبترجع False لو المقارنة دى غلط..

العلامة	المعامل Operator
==	يساوى (علامتين)
!=	لا يساوى
>	أكبر من
<	أصغر من
>=	أكبر من او يساوى
<=	أصغر من او يساوى

`>>> 5 == 5` `True`	هل 5 تساوى 5
`>>> 5 != 4` `True`	هل 5 لا تساوى 4
`>>> 5 < 6` `True`	هل 5 أقل من 6
`>>> 5 > 7` `False`	هل 5 أكبر من 7
`>>> 3 <= 5` `True`	هل 3 أقل من أو تساوى 5
`>>> 2 * 3 > 4` `True`	هل 2*3 أكبر من 4
`>>> 2*3+4*5 == 15*2-4` `True`	

- طبعاً خلى بالك لو كان أحد أطراف المقارنة عملية حسابية Math Operation العملية الحسابية ليها اولوية تسبق عملية المقارنة و هتتنفذ وبعدين هيقارن..



فاضل إنك تسأل فايدتها إيه عمليات المقارنة دى ؟! دى من أهم العمليات اللى بتتحكم فى سير البرنامج وهنستخدمها بشكل أساسى فى جمل الشرط والتكرار زى ما هنشوف قدام بس أصبر شوية.

العمليات المنطقية Logical Operations.

لو كنت قبل كده سمعت عن البوابات المنطقية Logic gates..ودى عبارة عن رمز بيعبر عن دالة بتاخد input او اكتر وتنفذ عليهم علاقة معينة ويطلع output واحد..وعرفت إن القيم المنطقية دى اما 0 أو 1 ده فى نظام البينارى طبعاً.

فى لغة البايثون أستفادوا من البوابات المنطقية دى وعملوا logical operators معاملات منطقية and و or و not .. والعوامل بتاعتها بتكون قيم True او False بدل 0 و 1 .. والخرج بتاعها بيكون True أو False حسب نوع المعامل and , or , not

False	And	False	False
True		False	False
False		True	False
True		True	False

False	or	False	False
True		False	True
False		True	True
True		True	True

والمعامل الأخير not ده بيعكس لو جاياله قيمة True هيخليها False والعكس..



تلاقيك بتسأل True و False والعمليات المنطقية دى هعمل بيها ايه ؟ 🤔

لو لسة فاكر عمليات المقارنة فهى اللى الناتج بتاعها بيكون True او False وبالتالى بالعمليات المنطقية نقدر نربط عمليات المقارنة مع بعض.. وهتعرف أهمية الموضوع ده قدام لما ناخد جمل الشرط .. بس أفتح الـ Interactive Interpreter دلوقتى و جرب معايا.

`>>> True and True` `True` `>>> True and False` `False` `>>> True or False` `True`	`>>> False or False` `False` `>>> not True` `False` `>>> not False` `True`

- لو العملية فيها معاملات حسابية ومقارنة ومعاملات منطقية هينفذ العملية الحسابية الأول وبعدين عملية المقارنة وبعدين العملية المنطقية..

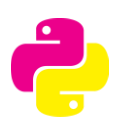
`>>> 5+3 < 4-1 or 3*4 == 2*6` `True`	`8<3    or 12==12` `False or  True   = True`
`>>> 5+2 > 4-3 and not 1 > 3` `True`	`7 > 1 and not False` `True  and   True     = True`
`>>> ((5+2)>(4-3))and not(1>3)` `True`	عشان متوهش فى العمليات تقدر تحط كل عملية بين أقواس والكود هيبقى واضح أكتر..



لو أحد العوامل اللى على يمين وشمال and أو or كان أرقام ففى الحالة دى ناتج العملية مش هيكون True او False.

- فى حالة المعامل and الناتج هيكون الرقم اللى على يمنيها.
- فى حالة المعامل or الناتج هيكون الرقم اللى علي شمالها.
- لو أحد العوامل كان صفر..فى حالة and الناتج طبعاً صفر.
- فى حالة or الناتج هيكون الرقم التانى.

`>>> 0 or 2` `2` `>>> True and 2` `2` `>>> 2 and True` `True`	`>>> 5 and 6` `6` `>>> 5 or 6` `5` `>>> 0 and 2` `0`



المتغيرات وأنواع البيانات

المتغيرات variables

المتغيرات هى أماكن فى الذاكرة بنخزن فيها قيم values (أرقام ونصوص) باسماء أحنا بنحددها عشان نقدر نستخدمها فى اكثر من مكان فى الكود أثناء تشغيل البرنامج.

تعريف المتغيرات Assigning Variables

تعريف المتغير أو بمعنى أخر إسناد قيمة للمتغير ... أزاى تخلى مفسر اللغة ياخد قيمة منك يحطها فى الميمورى باسم معين وانت تقدر تسترجع القيمة دى فى أى وقت أثناء تشغيل البرنامج.

```
variable_name = value
```

الطريقة سهلة .. هتكتب اسم المتغير وبعدين علامة = وبعدين القيمة اللى عاوز تخزنها فى المتغير ..

```
>>> x = 5
>>> x
5
>>> |
```

- مثلاً فى الكود اللى على شمالك قولنا للـinterpreter أحجز مكان فى الميمورى وسميه x وحط فيه القيمة 5
- دلوقتى لما أحتاج ارجع القيمة اللى جوة المتغير x اناديه باسمه وهيرجع القيمة على طول .



اسماء المتغيرات identifiers

فى لغة البايثون فيه حاجة اسمها identifier وده الاسم اللى بتسميه لمتغير variable او دالة function او كلاس class وهنعرف أخر أتنين معناهم ايه فيما بعد .. المهم الاسم بتاع المتغير أو الـ identifier مواصفاته ايه ؟ ..

- الـ identifier ممكن يكون أى حرف أو مجموعة حروف وأرقام بس لازم تبدأ بحرف أنجليزى أو عربى (لو شغال على IDLE) ومينفعش تحتوى على أى رموز زى %$#@ .
- الرمز الوحيد المسموح بيه هو _ underscore وده ينفع يكون اسم متغير لوحده.
- فى حاجة اسمها الـ reserved keywords أو الكلمات المحجوز فى اللغة ودى مينفعش تستخدمها كـ identifiers.
- وال keywords دى كلمات وجمل برمجية ليها استخدامات تانية فى اللغة زى جمل الشرط والتكرار تعريف الدوال والكلاسات وفيما بعد هتعرف أغلب الاسامى دى.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

جدول بالكلمات المحجوزة فى لغة بايثون



أمثلة على تعريف المتغيرات

>>> x = 2 >>> y = 3	هنا كتبنا 2 = x يعنى قولنا للمفسر روح خزن فى الرامه متغير اسمه x والقيمة بتاعته 2 . والسطر التانى عملنا متغير اسمه y وقيمته 3 .
>>> x 2 >>> y 3	بعدين كتبنا اسم المتغير أو عملناله استدعاء call يعنى قولنا للمفسر روح الرامه هاتلنا قيمة المتغير اللى اسمه x أو y .
>>> x*y 6 >>> x+y * x-y 5	بعدين بدأنا نستفيد بالمتغيرات دى إننا نستخدمهم فى العمليات الحسابية.

- اسم المتغير ممكن يكون _ ولو شغال على الـ IDLE ممكن يكون بالعربى.

اسناد متعدد للمتغيرات Multiple Assignment

يعنى ازاى فى سطر واحد تعرف اكتر من متغير..

var1,var2,var3=val1,val2,val3	var1=var2=var3 = val
>>> x,y,z=1,2,3 >>> x 1 >>> y 2 >>> z 3	>>> x=y=z=20 >>> x 20 >>> y 20 >>> z 20



- إسناد المتغير مش لازم يكون لقيمة صريحة ممكن يكون لـ متغير تانى أو ناتج عملية حسابية او منطقية أو دالة أو أوبجكت من كلاس.

```
>>> x = 1
>>> y = 2
>>> x = y
>>> y = x >= 2
>>> x
2
>>> y
True
```

أنواع البيانات Data Types

البيانات أو الـ data اللى بتتعامل معاها فى لغة البايثون أو اللى بتخزنها فى المتغيرات ممكن تقسمها إلى :

1- أرقام numbers

والأرقام دى وممكن تكون عدد صحيح integer يختصر بـ int زى 1 أو رقم كسر float زى 1.5 أو رقم معقد/تخيلى complex زى 5j

2- نصوص strings

أى حاجة بين علامتين ' ' أو " " زى جملة "Hello World" اللى كنا بنطبعها فى الأول دى اسمها نص string أو str

3- قيم منطقية Boolean

أو bool زى True أو False .. وخلى بالك إن الحرف الأول capital.

4- مصفوفات ودى 3 أنواع زى list , tuple , dictionary و هنتكلم عنهم قدام.



عشان تعرف النوع بتاع أى قيمة فى لغة البايثون بتستخدم دالة type

`>>> type(5)` `<class 'int'>` `>>> type(5.5)` `<class 'float'>` `>>> type(5j)` `<class 'complex'>`	`>>> type('Hello')` `<class 'str'>` `>>> type(True)` `<class 'bool'>`

نوع المتغير Variable type

نوع المتغير هو نوع الداتا اللى خزنتها فيه..

نوع البيانات من الحاجات المهمة لأنه هيفرق فى العمليات بين المتغيرات وكذلك فى الحجم اللى يشغله كل متغير فى الذاكرة.

```
>>> x = 1
>>> y = 1.5
>>> type(x)
<class 'int'>
>>> type(y)
<class 'float'>
```

البايثون لغة Dynamically typed يعنى نوع المتغير يتم تعرفه بشكل ديناميكى بناءاً على القيمة اللى بخزنها فيه, بالإضافة إني أقدر أغير نوع المتغير بتغيير القيمة اللى بخزنها فيه.

- هنا مثلاً المتغير x ممكن ياخد قيمه True أو False ويبقى نوعه bool او ياخد رقم ويبقى نوعه int او ياخد نص ويبقى نوعه str من غير أى مشاكل ..

```
>>> x = True
>>> type(x)
<class 'bool'>
>>> x=1
>>> type(x)
<class 'int'>
>>> x="Hello"
>>> type(x)
<class 'str'>
```



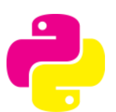
- بقولك كده ليه ؟. عشان لو أتعاملت مع لغات زى الـc/c++/java/c# المعروفة بأنها statically typed يعني نوع البيانات ثابت الوضع هيكون مختلف.
- لو عملت متغير بتحدد نوعه فى الكود ومينفعش تغير نوعه وقت التشغيل, ملثاً عاوز تعرف متغير رقمى صحيح اسمه x وقيمته 1 هتكتب int x = 1 وبعد ما عرفت المتغير مينفعش تخزن فيه قيم لأى نوع تانى غير الـinteger أو ارقام صحيحة.
- لكن لغة البايثون وفى الغالب اللغات المفسرة تعتبر Dynamically Typed

الأرقام Numbers

الأرقام زى ما قولنا ممكن تكون ارقام صحيحة integer وتختصر بـ int زى 1و2و100 أو ممكن تكون أرقام كسرية وتسمى float زى 0.7 أز 1.0 (لو حطيت 0. جنب الرقم الصحيح اللغة بتتعامل معاه أنه float)..أو ممكن تكون أرقام مركبة complex .

#التحويل بين الأرقام Number Type Conversion

`int(value)`	تحويل القيمة الى رقم صحيح
`float(value)`	تحويل القيمة الى رقم كسرى
`complex(value)`	تحويل القيمة الى رقم مركب
`complex(real,img)`	عمل رقم مركب جزء صحيح وأخر تخيلى



```
>>> int(5.5)
5
>>> float(10)
10.0
>>> complex(2,4)
(2+4j)
```

```
>>> x = 10
>>> type(x)
<class 'int'>
>>> x = complex(x)
>>> x
(10+0j)
>>> type(x)
<class 'complex'>
```

- دالة int بتحول أى قيمة رقمية إلى رقم صحيح
- دالة complex بتاخد برامترين الرقم الحقيقى real والرقم التخيلى imaginary وبتولد رقم مركب complex.

مكتبة Math

فى أغلب لغات البرمجة هتلاقوا مكتبة تحمل الاسم Math ودى تحتوى على دوال رياضية Mathematical Functions للعمليات على الأرقام ..

عشان تستخدم دوال المكتبة لازم تعملها import أو استدعاء بالطريقة دى وهنشرح معنى الطريقة دى فى شابتر الـ Modules..

```
from math import *
```

المكتبة فيها دوال كتير هنستخدم بعضهم.. و عشان نعرف الدوال نقدر نعملها print على الشاشة بدالة dir بالشكل ده ..

```
>>> from math import *
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
ma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```



`from math import *`	استدعاء كل الدوال من المكتبة math
`>>> pi` `3.141592653589793`	الرقم pi أو ط
`>>> sqrt(25)` `5.0` `>>> pow(2,4)` `16.0`	دالة الجزر sqrt دالة الأس pow .. هنا 2 أس 4
`>>> floor(5.6)` `5` `>>> ceil(5.6)` `6`	دالة floor بتجيب أقرب رقم صحيح أصغر من الرقم الكسري. دالة ceil بتجيب أقرب رقم صحيح أكبر.
`>>> log10(10)` `1.0` `>>> log(100,10)` `2.0`	دالة log10 بتجيب اللوغاريتم للأساس 10 دالة log بتاخد الرقم والأساس. هنا لوغاريتم 100 للأساس 10 = 2
`>>> factorial(4)` `24` `>>> fabs(-10)` `10.0`	دالة factorial بتجيب مضروب الرقم. هنا مضروب 4 = 4 * 3 * 2 * 1 = 24 دالة fabs بتجيب القيمة المطلقة للعدد.



`>>> round(10.12845,2)` `10.13` `>>> pi` `3.141592653589793` `>>> round(pi,2)` `3.14`	دالة round بتقرب الأعداد. بتاخد برامترين..الاول العدد اللى عاوز أقربه, والتانى عدد الأرقام الى تسيبهم بعد العلامة العشرية.
`>>> sin(90)` `0.8939966636005579` `>>> sin(radians(90))` `1.0` `>>> radians(180) == pi` `True` `>>> x=radians(90)` `>>> int(sin(x))` `1` `>>> int(cos(x))` `0`	الدوال المثلثية زى sin,cos,tan. فى لغة البايثون الدالة بتاخد قيمة الزاوية بالتقدير الدائرى radian. لو عملت sin(90) مش هتطلع 1 لأنك لازم تحول 90 إلى راديان بالدالة radians . طبعاً أنت عارف إن 180 بالتقدير الدائرى = باى

دى كانت بعض الدوال المهمة فى مكتبة Math .. لو عاوز تعرف استخدام باقى الدوال تقدر تشوفها على الوثيقة بتاعة اللغة math



النصوص Strings

من شوية كنا بنتكلم عن الأرقام بأنواعها..بس فيه نوع تانى من البيانات فى لغة البايثون وفى كل اللغات اسمه string وأختصارته str , وده أى حاجة تتكتب بين علامتين تنصيص من نوع single quotes او double quotes

```
>>> x = "Hello world"
>>> type(x)
<class 'str'>
>>> type('Hi')
<class 'str'>
>>> 'hi'
'hi'
>>> hi
NameError: name 'hi' is not
defined
>>>
```

النص لازم يتكتب بين علامات ' ' أو " "
لو كتبت النص من غير العلامات هيحصل إيرور وهيقولك not defined لأنه بالشكل ده أفتكره identifier لأى متغير أو دالة أنت عرفتها.. فدور فى المتغيرات وملقهوش فحصل الـ error.

```
>>> x = 5
>>> y = '5'
>>> type(x)
<class 'int'>
>>> type(y)
<class 'str'>

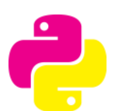
>>> x + y
TypeError: unsupported operand
type(s) for +: 'int' and 'str'

>>> x + int(y)
10
>>> str(x) + y
'55'
```

هنا عرفنا متغيرين الاول x نوعه int والتالى y ونوعه str

بعدين حاولت أجمع x مع y فحصل error لأنهم من نوعين اتاا مختلفين.

فكان الحل..إنك تحول المتغير y إلى متغير عددى وتبقى عملية حسابية.
أو تحول x إلى متغير نصى وتبقى عملية دمج بين نصين ويحطهم جنب بعض.



النص متعدد السطور Multi-Line String

لو عاوز نص فيه أكتر من سطر علامتين quote مينفعوش.. لازم تحط النص بين 3 علامات ..single quote /double quote

`>>> x = '''Hello` `World'''` `>>> x` `'Hello\nWorld'` `>>> print(x)` `Hello` `World`	`>>> x="""1` `2` `3"""` `>>> x` `'1\n2\n3'` `>>> print(x)` `1` `2` `3`

- خلى بالك لما كتبت اسم المتغير جابلك شكل النص المتخزن فى الميمورى وياريت تكون ملاحظ إن العلامة \n تعبر عن سطر جديد..والعلامات دى اسمها escape character وهى حروف ليها استخدمات مختلفة داخل النصوص هتكلم عليها الفقرة الجاية.
- لما عملت print للمتغير ظهرت السطور.

الحروف الخفية escape character

مش عارف أضربلها ترجمة بالعربى 😂 بس الحروف الخفية وصف مناسب ليها لأنها حروف بتتكتب فى النص بعد علامة slash \ ومش بتتطبع non printable لكنها بتأدى وظيفة تانية.



```
>>> x = "Hello\nWorld"
>>> x
'Hello\nWorld'
>>> print(x)
Hello
World
```

```
>>> x = "Hello\tWorld"
>>> x
'Hello\tWorld'
>>> print(x)
Hello    World
```

- الحرف الأهم هو \n وده بيكتب الكلام اللى بعده فى سطر جديد.
- لاحظ على الشمال.. لما كتبنا كلمة Hello\nWorld والحرف \n متطبعش لما عملت print ولكنه خلى اللى بعده يظهر فى سطر جديد.
- تانى حرف \t بيعمل tab او بيدى مسافة بين اللى قبله واللى بعده..

أنت عارف إننا بنكتب النص بين علامات double/single quotes.. طيب لو عاوز تكتب علامة منهم جوة النص هيحصل ايه؟!🤔

```
>>> x = "Hello"world"
SyntaxError: invalid syntax
>>> x = 'Hello'world'
SyntaxError: invalid syntax
```

فى الحالة دى "Hello"world" لما شاف double quote أفتكر إنك قفلت النص بيها وإن نهايته كلمة Hello , واللى بعدها ده حاجة غير معروفة فحصل إيرور..



الحل إنك تحط قبل الـ quote علامة slash فتكتبها كده "\

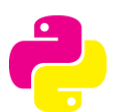
أو حل تانى لو عاوز تحط single quote جوة النص..حطه بين double quotes والعكس.

```
>>> x = "Hello\"world"
>>> x
'Hello"world'
>>> x = "Hello'world"
>>> x
Hello'world
```

باقى الحروف الخفية أو تقدر تشوفهم من هنا strings

Escape Sequence	Meaning
newline	Ignored
\\\\	Backslash (\\)
\\'	Single quote (')
\\"	Double quote (")
\\a	ASCII Bell (BEL)
\\b	ASCII Backspace (BS)
\\f	ASCII Formfeed (FF)
\\n	ASCII Linefeed (LF)
\\r	ASCII Carriage Return (CR)
\\t	ASCII Horizontal Tab (TAB)
\\v	ASCII Vertical Tab (VT)
ooo	ASCII character with octal value *ooo*
\\x*hh...*	ASCII character with hex value *hh...*

أغلب الحروف الخفية مش هتعرف تشوف بيعملوا أيه فى برنامج IDLE لازم تفتح البايثون فى الـ command line.



العمليات على النصوص String Operation.

شوفنا العمليات على الأرقام زى العلميات الحسابية وغيرها.. النصوص برده بتدعم بعض العمليات..

دمج النصوص Concatenation

- المعامل + بيدمج نصين أو اكتر بيحط النص الأول جنب التانى.
- لازم يكون المتغيرين x و y من نفس النوع str عشان ميحصلش خطأ.

```
>>> x = "Hello"
>>> y = "World"
>>> z = x + y
>>> z
'HelloWorld'
```

تكرار النص Repetition

- علامة * الضرب بتكرر النص عدد من المرات
- السطر التانى مثال للدمج والتكرار..

```
>>> x * 3
'HelloHelloHello'
>>> x * 2 + " : " + y * 2
'HelloHello : WorldWorld'
```

أقتطاع النص Slicing

- لغة البايثون بتتعامل مع النص كمصفوفة مكونة من حروف كل حرف ليه index أو رقم والترتيب بيبدأ من صفر..مثلاً كلمة Hello ده شكل الـ index بتاع كل حرف فيها.

H	E	L	L	O
0	1	2	3	4



- عشان ترجع حرف من النص بتكتب اسم المتغير النصى وبعده أقواس مربعة وجواه الـindex بتاع الحرف. str[index]

```
>>> x
'Hello'
>>> x[0]  #رجع الحرف الأول
'H'
>>> x[0:2]#من الحرف الاول خد حرفين
'He'
>>> x[3:] #خد من الحرف الرابع للأخر
'lo'
```

- لو عاوز ترجع جزء معين بتكتب بداية الجزء ده وعدد الحروف اللى هتاخده بداية منه .str[index:length]

أدخال متغيرات داخل النص Format

هتشوف قدام أهمية الموضوع ده ممكن متحساش دلوقتى .. بس لو عندك نص عاوز تدخل جواة قيمة متغير عندك حلين :

- الأول تعمل concatenation بأنك تحط جزئ من النص وبعدين علامة + وبعدين المتغير وبعدين + وبعدين باقى النص
- مثلاً عاوز أدخل قيمة المتغير x داخل جملة the number is .. students فهيكون بالشكل ده.

```
>>> x = 120
>>> "the number is " + str(x) + " student"
'the number is 120 student'
```

- بس شايف استخدمنا كام علامة تنصيص ودالة str اللى تحول المتغير الرقمى إلى نصى عشان ينفع يتجمع مع النص..الموضوع رخم بعض الشئ..
- الحل التانى تستخدم الـ formatting operator .. كل اللى هتعمله إنك هتحط المعامل %s جوه النص فى المكان اللى عاوز تدخل بعد قيمة المتغير.



- وبعده علامة quote اللى بتقفل النص تحط علامة ٪ والمتغيرات بعدد وترتيب علامات %s اللى حطتهم جوة النص.

```
>>> x = 120
>>> "the number is %s students"%x
'the number is 120 students'
>>> name,age,phone='Mahmoud',23,'011'
>>> x="Name : %s\nAge : %s\nPhone : %s"%(name,age,phone)
>>> print(x)
Name : Mahmoud
Age : 23
Phone : 011
```

دوال النصوص String functions

لغة البايثون فيها دوال مهمة للعمل على النصوص هشرح بعضها وأوضح بتعمل ايه

>>> x = "Pink Python" >>> len(x) 11	دالة len بتجيب طول النص أو عدد حروفه. هنا الدالة أعتبرت المسافة الفاضية حرف.
>>> x.upper() 'PINK PYTHON' >>> x 'Pink Python' >>> x = x.upper() >>> x 'PINK PYTHON' >>> x.lower() 'pink python' >>> x 'PINK PYTHON'	دالة upper بتحول النص إلى حروف كبيرة capital. ودالة lower بتحولهم لحروف صغيرة small. خلى بالك دالة upper أو lower لو طبقناها على متغير نصى بترجع نسخة من النص اللى فى المتغير مش بتعدل على المتغير نفسه. لو عاوز أحول قيمة المتغير نفسه لازم أعمله assignment بالقيمة اللى بترجعها الدالة var = var.upper()



<pre>>>> x 'PINK PYTHON' >>> x.find('k') -1 >>> x.find('K') 3</pre>	دالة find بتاخد برامتر واحد و هو الحرف أو الكلمة اللى عاوزين ندور عليها فى النص وبترجع الـ index بتاع الحرف لو موجود..ولو مش موجود بترجع 1-. فى الاول هنا الدالة رجعت 1- عشان k مش موجود لكن الحرف الموجود هو K كابتال.
<pre>>>> x 'PINK PYTHON' >>> x.startswith('P') True >>> x.endswith('n') False >>> x.endswith('N') True >>> x.endswith('PYTHOn') False >>> x.endswith('PYTHON') True</pre>	دالة stratswith بتاخد برامتر واحد وهو الحرف أو الكلمة اللى عاوزين نعرف إذا كان النص يبدأ بيها.. وبترجع True أو False. دالة endswith بتشوف إذا كان النص بينتهى بحرف او كلمة.

وبرده لو عاوز تعرف أكتر عن النصوص تقدر تكمل من هنا الموقع الرسمى من هنا string



المصفوفات Arrays

فى أى لغة برمجة المصفوفة هى مجموعة من العناصر أو القيم اللى من نفس النوع أو من أنواع data types مختلفة بيتم تخزينهم جنب بعض فى الميمورى باسم او identifier واحد ويتم أسترجاع كل عنصر بترتيبه أو الـ index بتاعه فى المصفوفة.

فى لغة البايثون بتختفى كلمة مصفوفة أو array ويظهر أكتر مصطلح أو نوع لسلاسل البيانات أو الـ data sequence ليهم نفس الفكرة... وهنتكلم عن الأنواع المشهورة زى list و tuple و dictionary.

الليستة List

الـ list ده نوع من البيانات بيتيح ليك إنك تخزن أكتر من قيمة أو عنصر من نوع واحد او أنواع مختلفة.

طريقة تعريفها List_name = [data , data , data] حيث data ممكن يكون قيمة رقمية number أو string أو bool أو أى object فى اللغة.

`>>> x = [1,'Hi',True]` `>>> type(x)` `<class 'list'>`	هنا عرفنا list اسمها x فيها 3 عناصر مختلفة.
`>>> x[0]` `1` `>>> x[1]` `'Hi'` `>>> x[2]` `True`	عشان تطلع عنصر من الـ list بيكون كده list_name[index]. وخلى بالك إن الـ index بيبدأ من 0 مش 1 مثلاً عشان تطلع العنصر الأول الإندكس بتاعه 0 والعنصر التانى 1 وهكذا..



`>>> x = [0,1,2,3,4,5]` `>>> x[0:1]` `[0]` `>>> x[0:3]` `[0, 1, 2]` `>>> x[1:3]` `[1, 2]` `>>> x[3:5]` `[3, 4]` `>>> x[1:4]` `[1, 2, 3]`	عشان تطلع عدد محدد من العناصر بيكون بالشكل ده List_name[start:start+length] مثلا فى الليستة دى [0,1,2,3,4,5] وأطلع العناصر 1و2و3.. إندكس العنصر 1 هو البداية 1 و عدد العناصر 3 يبقى النهاية عند 1+3 = 4 x[1:4]
`>>> y = ['a','b','c']` `>>> x = [1,True,y]` `>>> x` `[1, True, ['a', 'b', 'c']]` `>>> x[2]` `['a', 'b', 'c']` `>>> x[2][0]` `'a'` `>>> type(x[2])` `<class 'list'>`	ممكن العنصر اللى فى الـ List برده يكون list. هنا y ليستة فيها 3 عناصر..حطينا y كعنصر فى الليستة x . عشان تطلع العنصر الأول (إندكس 0) من الليستة الداخلية اللى الإندكس بتاعها 2 فى x بالشكل ده x[2][0] كان ممكن تطلع الليستة الداخلية فى متغير xx=x[2] وبعدين تطلع العنصر من المتغير الجديد. xx[0]

بعض الدوال على الليستات

`>>> x = [1,2,4,-2,4,1,5]` `>>> len(x)` `7` `>>> max(x)` `5` `>>> min(x)` `-2`	دالة len بتحسب عدد العناصر. دالة max بتطلع أكبر عنصر دالة mix بتطلع أصغر عنصر.
`>>> x.append(10)` `>>> x` `[1, 2, 4, -2, 4, 1, 5, 10]`	دالة append بتضيف عنصر فى أخر الليستة.



`>>> x` `[1, 2, 4, -2, 4, 1, 5, 10]` `>>> x.insert(0,-5)` `>>> x` `[-5, 1, 2, 4, -2, 4, 1, 5, 10]` `>>> x.insert(4,-9)` `>>> x` `[-5, 1, 2, 4, -9, -2, 4, 1, 5, 10]`	دالة insert بتضيف عنصر في إندكس معين في الليسته. بتاخد برامترين الأول الإندكس اللى هضيف فيه..والتاني العنصر اللى هضيفه.
`>>> x.reverse()` `>>> x` `[10, 5, 1, 4, -2, -5, 4, 2, 1, -5]`	دالة reverse بتعكس ترتيب عناصر الليسته.
`>>> x.sort()` `>>> x` `[-5, -5, -2, 1, 1, 2, 4, 4, 5, 10]` `>>> x.sort(reverse=True)` `>>> x` `[10, 5, 4, 4, 2, 1, 1, -2, -5, -5]`	دالة sort بتعمل ترتيب تصاعدي لعناصر الليسته. لو عاوزها تعمل ترتيب تنازلي بتديها برامتر sort(reverse=True)
`>>> x` `[10, 5, 4, 4, 2, 1, 1, -2, -5, -5]` `>>> x.index(2)` `4`	دالة index بتاخد برامتر بالعنصر وبترجع مكانه في الليسته.
`>>> x` `[10, 5, 4, 4, 2, 1, 1, -2, -5, -5]` `>>> x.count(-5)` `2`	دالة count بترجع مرات تكرار العنصر في الليسته.



الصفوف tuples

النوع التانى من المصفوفات وهو شبيه بالـ list ويسمى tuples ويوجد أكتر من فرق بينه وبين الليست..

- طريقة تعريف الـ list بستخدم أقواس مربعة

```
list_name = [data,data,data]
```

- طريقة تعريف الـ tuples بستخدم أقواس دائرية

```
tuple_name = (data,data,data)
```

- طرقة استخراج الداتا من الـ list و tuple واحده

```
list_name [index]
tuple_name [index]
```

يعنى الفرق فى الشكل بس🤔؟! لا العناصر فى ال tuples غير قابلة للتعديل يعنى بعد ما تعرفها متقدرش تعدل عليها.

```
>>> x = [1,2,3]
>>> y = (1,2,3)
>>> type(x)
<class 'list'>
>>> type(y)
<class 'tuple'>
>>> x
[1, 2, 3]
```



```
>>> x[0]=-1
>>> x
[-1, 2, 3]
>>> y
(1, 2, 3)
>>> y[0]=-1
TypeError: 'tuple' object does not support item assignment
```

القاموس Dictionary

النوع التالت هو الـ dictionary أو ممكن تقول عليه القاموس .. مش مجرد مجموعة عناصر مرصوصة جنب بعض وتقدر توصلهم عن طريق الـ index ولكن هنا كل عنصر ليه اسم تقدر توصله من خلاله.

طريقة التعريف..لاحظ بنستخدم أقواس مجموعة أو curly brackets { }

```
dict_name = { key1 : value , key2 : value , key3 : value }
```

وعشان أطلع الـ value المقابلة للـ key معين بالشكل ده.

```
dict_name['key_name']
```



ودى أمثلة على استخدام الـ dicts.

```
>>> x = {'name':'Mahmoud','age':23,'country':'Egypt'}
>>> type(x)
<class 'dict'>
>>> x['name']
'Mahmoud'
>>> x['age']
23
>>> x['country']
'Egypt'
>>> x[0]
KeyError: 0
>>> x['Age']
KeyError: 'Age'
```

- لاحظنا فى الأمر الاخير حاولنا نستخرج قيمة المفتاح Age ونتج عنها إيررور KeyError لأن الـkey متسجل باسم age اول حرف small وده نفس الحال لما تستدعى متغيرات عاديه لأن اللغة case sensitive.

`>>> x = {'color1':'red', 'color2':'green','color3':'blue'}` `>>> x['color1']` `'red'` `>>> x.get('color1')` `'red'`	عشان تطلع value من الـ dict اما عن طريق الأقواس المربعة او عن طريق داله get. `dict.get('key')` `dict['key']`
`>>> x['color3']='yellow'` `>>> x` `{'color1': 'red', 'color2': 'green', 'color3': 'yellow'}`	عشان تحدث قيمة value فى الـ dict بتعملها assignment بالـkey عادى



`>>> x['color4']='orange'` `>>> x` `{'color1': 'red', 'color2': 'green', 'color3': 'yellow', 'color4': 'orange'}`	عشان تضيف key و value جداد في الـ dict بتعمل برده assignment ولو الـ key مش موجود هيضيفه ولو موجود هيحدث القيمه.
`>>> x.pop('color4')` `'orange'` `>>> x.pop('color3')` `'yellow'` `>>> x` `{'color1': 'red', 'color2': 'green'}`	عشان تسمح key من الـ dict بتستخدم الدالة pop اللى بتاخد براميتر باسم الـ key
`>>> 'color1' in x` `True` `>>> 'color3' in x` `False`	عشان تتأكد إن الـkey موجود في الـ dict بتستخدم الامر in اللى بيرجع True أو False.
`>>> list(x.keys())` `['color1', 'color2']` `>>> list(x.values())` `['red', 'green']`	عشان تطلع كل الـ key او الـ values من الـ dict بنستخدم الدوال keys() و values().. وطبعاً الدالة list بتخزن القيم دى في لستة

وتقدر من هنا تعرف أكتر عن المصفوفات او الـ data structures في لغة البايثون.



التعليقات Comments

فى أى لغة برمجة.. التعليق ده كلام أنت بتكتبه فى الكود عشان تشرح الأمر أو الجزء ده وظيفته أيه ولكنه لا يعتبر جزء من الكود بتاعك او الـ interpreter مش بيشوفه.

و فى لغة البايثون فيه نوعين من التعليقات.

- تعليقات داخل السطر وهو أى كود تحط قدامه علامة شباك hash #

```
# single line comment
```

- النوع التانى تعليق متعدد الأسطر وده بتحوله الكود أو الكلام اللى عاوز تخليه تعليق لنص string بأنك تحط الكود بين 3 علامات single/double quotes والمفسر لما يشوفه مش هيعمله حاجة.

```
"""
comment1
Comment2
Comment3
Comment..
"""
```

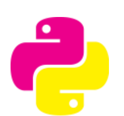
```
'''
comment1
Comment2
Comment3
Comment..
'''
```



السكريت ده بيوضحلك طريقة عمل كومنت سطر واحد ومتعدد الأسطر وكومنت جوة السطر .. وطبعاً الكلام اللى ملون بالأحمر دى هى الكومنتات واللى مش هيشوفها الإنتربريتر.

<pre>#Hi! I'm A comment. #the Interpreter will not see me ^_^ print(1) #print(2) print(3); print(4); #print(5) #for Multiline comment #Turn it to a string ''' print('you') print("can't") print('see') print('Me :3') '''</pre>	لما تعمل run للكود ده الخرج هيكون كده 1 3 4

download script comment.py from GitHub



الدوال Functions

سمعت اكتر من مرة كلمة دالة function وشوفنا اوامر برمجية كتير قولنا عليها دوال زى الدالة اللى بتطبع حاجة على الشاشة print ودى الدال اللى أستخدمناهم مع الـ list و tuple و dict..

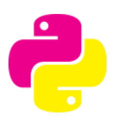
لو فاكر الدالة فى الرياضة..فهى حاجة بتاخد معطيات inputs وبتنفذ عليها علاقة معينة وتطلع الخرج output.

فى لغة البايثون أو أى لغة برمجة..الدالة بتاخد معطيات (متغيرات / قيم / data) وبتنفذ عليهم كود معين وبترجع أو مترجعش نتيجة حسب نوع الدالة دى .

الدوال اللى بتتعامل معاها فى اى لغة برمجة نوعين.

- النوع التانى اللى هنتكلم عنه بعدين اسمها user defined functions يعنى أنت اللى بتعملها وهنعرف بتعملها ليه و ازاى..
- النوع الأول بقى مش أنت اللى بتعملها ويادوب بتستخدمها عشان تنفذ بيها أوامر معينة والدوال دى اسمها built-in أو دوال تابعة للغة نفسها زى دالة print و min و max و len وtype
- الدوال دى معملناش حاجة غير إننا أستخدمناها بعتنا ليها داتا ونفذت عليها أوامر معينة..ومنعرفش الدالة دى عملت كده ازاى وميهمناش غير النتيجة بتاعتها.

وهنا جدول وبكل الدوال الـ built-in functions التابعين للغة البايثون تقدر تشوفهم وهتعرف إنك أستخدمت عدد كبير منهم لحد الآن.



- ممكن تكون لاحظت إن فيه دوال بقدر أستخدمها على طول ودوال تانيه تابعة لمتغيرات أو datatypes معينة.. مثلاً عندى ليسته x عشان أشوف عدد عناصرها بستخدم دالة len(x) وبديها اسم الليسته. ولو عاوز أعملها reverse بكتب x.reverse() طيب أيه الفرق 🤔؟

```
>>> x = [1,2,3]
>>> len(x)
3
>>> x.reverse()
>>> x
[3, 2, 1]
```

- دالة len من الدوال built-in functions بتقدر توصلها بشكل مباشر مش تابعة لأى موديول أو كلاس.
- أما دالة reverse مثلاً دى تابعة للمتغير x وهنا هسبق الأحداث وأقولك إن x مش متغير عادى , ده اسمه object أو كائن للـ class اسمها List.
- والكلاس دى مجموعة من الدوال والمتغيرات بتنفذ وظائف معينة ودوال الكلاس List دى أتعاملنا مع بعضها فى الجزء بتاع الـ list زى count و index و pop و append وعشان توصل للدوال بيكون عن طريق الـ object أو المتغير اللى بيمثل الكلاس دى وهو الليسته اللى أنت عملتها.

استخدام الدوال call function

لما تستخدم أى دالة أياً كان نوعها ده اسمه أنك بتعملها استدعاء call.. وعشان تستدعى الدالة دى بتكتب اسمها متبوعاً بأقواس بداخلها الداتا أو المعطيات اللى هتبعتها للدالة واللى بيسموها arguments وبتبقى متغيرات أو قيم مفصول بينها بـ comma.

فيه دوال بترجع نتيجة return type فممكن تاخد الخرج بتاعها فى متغير وفيه دوال مش بترجع حاجة مثلاً زى دالة print.



`function_name(arg1,arg2,arg…)`
`variable=function_name(arg1,arg2,arg…)`

`>>> x = print("Hello")` `Hello` `>>> x`	هنا الدالة print أخد argument عبارة عن نص وطبعته على الشاشة ومرجعتش حاجة فى المتغير x .
`>>> x = len("Hello")` `>>> x` `5`	هنا الدالة len أخدت argument قيمة نصية ورجعت عدد حروفها فى متغير x

- بالنسبة للمتغيرات اللى بتمررها للدوال وقولنا أنها تسمى arguments وممكن تسمع ليها مسمى parameter وده المتغير اللى بيكون فى تعريف الدالة نفسها وهتعرف ده اكتر لما أشرحلك الدوال user defined functions .. المهم أى قيم أو متغيرات هنديلها للدوال هسميها برامترز للتبسيط 😁.



دوال الإدخال والاخراج Input and Output

تمكن وانت وبتتعامل مع الـ interpreter التفاعلى بتكتب أى حاجة وبيطلعلك الناتج او بتكتب اسم المتغير فيظهر قدامك على الشاشة ومفيش مشاكل...لكن لو بتكتب سكريت إنت مش متفاعل مع الـ interpreter ومحتاج تطبع على الشاشة قيم المتغيرات والنواتج اللى بيطلعها الكود بتاعك فبتحتاج دوال المخرجات output زى دالة print

```
#print دالة
print('Hi') #بتطبع نصوص
print(123)  #أرقام

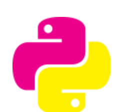
x = [1,2,3]
print(x)     #list
y = {'day1':'sat','day2':'sun'}
print(y)     #dict

name = "Mahmoud"   #متغير نصى
#عشان نطبع نص جنب المتغير
#خلى المتغير برامتر تانى فى الدالة
print("1-My Name is:",name)

#أو نعمل دمج بين النص والمتغير النصى بعلامة +
print("2-My Name is:"+name)
age = 23  #متغير عددى
#ينفع نمرره كبرامتر تانى وهيتطبع جنب النص
print("3-My Age is:",age)
#أو نستخدم المعامل %
print("4-My Age is: %s"%age)
#مينفعش ندمج بين متغير نصى ومتغير عددى وهيطلع ايرور
print("4-My Name is:"+age)
```

```
Hi
123
[1, 2, 3]
{'day1': 'sat', 'day2': 'sun'}
1-My Name is: Mahmoud
2-My Name is:Mahmoud
3-My Age is: 23
4-My Age is: 23
Traceback (most recent call last):
print("4-My Name is:"+age)
 TypeError: must be str, not int
```

Download script print.py from GitHub



دالة الأدخال input

لحد الآن احنا بنكتب كل القيم والمدخلات بتاعتنا فى الكود ونعمل run فتشتغل.. عاوزين أثناء تشغيل البرنامج المستخدم هو اللى يدخل القيم وهو يخزنها فى متغيرات ويعمل عليها علمياته ومعالجاته ويطبعها... وده بيتم بدالة اسمه input وشكلها بيكون كده..

دالة input بتاخد برامتر واحد وهو الرسالة اللى هيعرضها للمستخدم مثلاً تقوله أكتب نص معين .. وبعدين هترجع القيمة اللى هيكتبها المستخدم ويدوس enter على شكل متغير نصى str وخلى بالك من نوع المتغير..

```
var=input()
var=input('Message')
```

ده مثال على استخدام دالة input ..

```
>>> x = input()
Hi
>>> print(x)
Hi
>>> x = input('write something : ')
write something : Hello ^_^
>>> print(x)
 Hello ^_^
```

- فى ملاحظة مهمة لازم تعرفها على دالة input..إن القيمة اللى بترجعها بتبقى على شكل نص string حتى لو اللى دخلته كان رقم.
- النقطة دى تفرق معاك لو عاوز تستخدم المدخل ده فى عملية حسابية فلازم تحوله إلى متغير عددى int او float..



```
>>> x=input('Enter A number : ')
Enter A number : 10
>>> type(x)
<class 'str'>
>>> x + 5
TypeError: must be str, not int
>>> int(x) + 5
15
```

تعالو نستفيد بدالة الإدخال input ودالة الاخراج print ونعمل سكريت آله حاسبة بسيط يطلب من اليوزر إنه يدخل الرقم الاول ويدخل الرقم التانى وبعدين ينفذ عليهم العمليات الحسابيه .

```
print("HI ^_^")

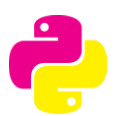
#أدخل الرقم الأول
num1 = input("Enter The First Number : ")
#حول المتغير النصى إلى عددى
num1 = float(num1)

#أدخل الرقم الثانى
#واحوله إلى متغير عددى فى نفس السطر
num2 = float(input("Enter The Second Number : "))

#ينفذ على المتغيرين العمليات الحسابية
print("num1 + num2 = " + str(num1 + num2))
print("num1 - num2 = " + str(num1 - num2))
print("num1 * num2 = " + str(num1 * num2))
print("num1 / num2 = " + str(num1 / num2))
```

```
HI ^_^
Enter The First Number : 20
Enter The Second Number : 5
num1 + num2 = 25.0
num1 - num2 = 15.0
num1 * num2 = 100.0
num1 / num2 = 4.0
```

Download input_output.py from GitHub



دوال المستخدم User Defined Functions

شغالين من بدرى بنتكلم عن الدوال وعرفنا إن فيه دوال built in functions دى تابعة للـ interpreter بتاع اللغة وأتكتبت معاه..منعرفش الكود اللى جواها بيعمل ايه بس يهمنا نعرف اسمها ونعرف نستخدمها ازاى.

فى نوع تانى أحنا بنعمله وبنحدد اسمه والبرامترات بتاعته وهنعرف بنعمله ليه..

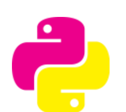
تعريف الدالة : هى عبارة عن مجموعة من الجمل البرمجية يتم تنفيذها فقط عندما يتم استدعاءها..ويمكن استدعاء فى أى موضع من الكود.

هوضحلك معنى التعريف ده قدام شوية بس تعالى نشوف أزى بنعمل الدوال.

تعريف الدالة Function definition.

```
def name(parameter1, parameter2,parameter n):
    code…
    code…
```

- بتستخدم كلمة def عشان تقول للـ interpreter إن اللى هتشوفه بعد كده دى دالة.
- اسم الدالة تنطبق عليه مواصفات الـ identifier انه مينفعش يكون فيه رمز أويبدأ برقم وميكونش تبع الـ keywords زى ما عرفنا فى المتغيرات.
- والبرامترات دى الأماكن اللى همرر قيم ومتغيرات للدالة من خلالها.
- بعد ما اقفل الكود بحط علامة colon دى بتعرف للـ interpreter إن اللى جاى هو بلوك من الكود تابع للدالة دى.



البلوكات Code Blocks.

قولنا الدالة لما أستدعيها بتنفذ مجموعة من الجمل البرمجية وعشان المفسر يعرف إن الأكواد دى خاصة بالدالة بتاعتنا بتتحط بنظام محدد اسمه block.

شكل الـ block بيختلف من لغة برمجة لتانية .. مثلاً اللغات عائلة الـ c/c++ بيستخدموا اقواس المجموعة curly brackets وفيه لغات تانية بتستخدم جملة start فى الأول وبعدين الأكواد بتاعة البلوك وبعدين جملة end وهكذا.. لو عاوز تعرف أكتر عن البلوكات من هنا Block (programming).

لغة البايثون اللى تهمنا دلوقتى بتستخدم طريقة الـ line indent... يعنى المسافة من بداية السطر ... بأختصار مجموعة الجمل البرمجية اللى عاوز أخليها فى بلوك بخليها على محاذاة واحدة بمسافة ثابتة عن بداية السطر والموضوع ده مهم.

تعالى ناخد مثال عشان متوهش.. هنعمل سكريبت ونعرف فيه دالة اسمها xx عاوزينها تطبع الأرقام من 1 الى 5

<pre>def xx(): print(1) print(2) print(3) print(4) print(5)</pre>	جمل print الخمسة هم البلوك بتاع الدالة xx لو عملت run للكود على كده مش هيظهر حاجة لازم تستدعى الدالة باسمها. xx()



def xx(): print(1) print(2) print(3) print(4) print(5) xx()	1 2 3 4 5

def xx(): print(1) print(2) print(3) print(4) print(5) xx()	لو جملة وحده فى الكود مكنتش بنفس محاذاة البلوك هيحصل error.

xx() def xx(): print(1) print(2) print(3) print(4) print(5)	لو عملت call للدالة قبل ماتعرفها هيطلعلك undefined error لأن للحظة دى الـ interpreter مبيعرفش يعنى ايه xx NameError: name 'xx' is not defined



من مـمـيزات الـدوال إنها بتخلـى الكـود reusable يعنـى هيكـون عـندك أكـتر من جـملة برمجية بدل ما تكـتبـهم أكتر من مرة فى الســكـريبت والـكـود يطـول, هتحـطـهم فى دالة وتقـدر تسـتدعـى الـدالة دى فى أى وقت وده الـهـدف الأهـم من اسـتـخـدام الـدوال.

`def xx():` `    print(1)` `    print(2)` `    print(3)` `    print(4)` `    print(5)` `print('call xx')` `xx()` `print('call again')` `xx()` `print('and again')` `xx()`	`call xx` `1` `2` `3` `4` `5` `call again` `1` `2` `3` `4` `5` `and again` `1` `2` `3` `4` `5`

الـدوال الـلى بتـرجـع قيـم return type

فاكـر دوال max و len كـانوا بيرجـعـوا قيـم نقـدر نخزنها فى متـغـير.. ونفـس الـكـلام مع الـدوال الـلى بنـعرفـها نقـدر نخليها ترجع قيـمة ونخزنها فى متـغير, والـنوع ده أسـمـه return type يعنـى دالة بـترجـع نتيـجـة..الـفـرق الـوحيـد إنك فى نـهاية الـبلـوك بتـضيـف كـلـمة return وقدامها المتغير أو الـقيـمـة الـلى عـاوز الـدالـة ترجـعـهـا.

```
def name(parameter1, parameter2,parameter n):
    code…
    code…
    return variable...
```



ده مثال على دالة اسمها add الدالة بتاخد برامترين num1 و num2 وتجمعهم وترجع مجموعهم ونقدر نخزن مجموعهم فى متغير.

<pre>def add(num1,num2): result = num1 + num2 return result x = add(5,6) print(x) y = add(10,30) print(y) print(add(100,300))</pre>	هنا عرفنا دالة اسمها add بتاخد برامترين num1 و num2 وبتجمع المتغيرين وتخزنهم فى المتغير result وترجع قيمة المتغير ده. 11 40 400 >>>

Download script functions1.py from GitHub

- لاحظ أهمية جملة return فى اخر بلوك الدالة هى اللى بتخلى الدالة ترجع قيمة المتغير result واللى هو مجموع الرقمين.
- دالة print() مش شرط البرامترات بتاعتها تكون قيم أو متغيرات بس ممكن تكون داول من نوع return type ولما تحطها كبرمتر جوه الدالة , هيتعمل call للدالة وتتنفذ وترجع الـ result ويتم طباعته بدالة print عادى جداً.



من الملاحظات اللى لازم تعرفها وانت وبتمرر متغيرات للدالة كـ parameters لغة البايثون بتديك أوبشنين لترتيب البرامترات دى.

- الأول إنه يكون ترتيبها فى التعريف وتسمى parameters هو ترتيبها فى الاستدعاء وتسمى arguments من الآخر يعنى المتغير قيمته هتروح للبرامتر المقابل ليه وده شئ بديهى ومعروف.

```
def name(par1,par2,par3,par4):
    code…

name(arg1,arg2,arg3,arg4)
```

أدينى ملونلك الـ argument و الـ parameters ومزبطلك 😁 عشان تعرف إن كل قيمة هتروح للمتغير المقابل وه شىء عادى ومتوقع.

- الطريقة التانيه إنك فى الاستدعاء تحدد البرامتر اللى عاوز تبعتله قيمة ومش لازم بالترتيب كمان

```
def name(par1,par2,..):
    code…
name(par1=arg1,par2=arg2,..)
```

شوف المثال ده هيوضحلك الفرق فى الطريقتين

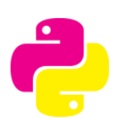


<pre>def xx(x,y,z): print("x = %s , y = %s , z =%s"%(x,y,z)) x=1 y=2 z=3 xx(x,y,z) xx(x=x,y=x,z=x) xx(z,y,x) xx(z=z,y=y,x=x)</pre>	<pre>x = 1 , y = 2 , z =3 x = 1 , y = 1 , z =1 x = 3 , y = 2 , z =1 x = 1 , y = 2 , z =3</pre>

Download script functions2.py from GitHub

المثال ده فيه أكتر من فكرة لازم تخلى بالك منهم.

- المتغيرات x,y,z اللى برة الدالة مالهمش علاقة بالبرامترات x,y,z اللى جواها والفقرة الجاية هنعرف الفرق بينهم.
- أول استدعاء للدالة xx(x,y,z) كل متغير هتتمر القيمة بتاعتها للبرامتر المقابل xx(1,2,3) فالـ interpreterهيشوفها كده xx(x=1,y=2,z=3) مفيش مشاكل..
- تانى استدعاء xx(x=x,y=x,z=x) اللى لونهم احمر دول البرامترات اللى جوة الدالة. هيبقى كده xx(x=1,y=2,z=3) يعنى انا حددت على طول قيمة كل برامتر ومكنتش محتاج ألتزم بترتيب البرامترات زى ما عملت فى الاستدعاء الأخير وعملت كده xx(z=z,y=y,x=x) فكانت xx(z=3,y=2,x=1) برده قيمة x,y,z متغيرتش ومعتمدتش على الترتيب.
- اللى أتغير فى الاستدعاء التالت xx(z,y,x) اللى قيم المتغيرات فيه xx(3,2,1) فالإنتربريتر هيمرر القيم بالترتيب كل قيمة للبرامتر المقابل فهتبقى xx(x=3,y=2,z=1).



نطاق المتغيرات Scope of Variables

شوفت فى المثال اللى فات إننا عملنا متغيرات x,y,z ودول بره الدالة وبرامترات x,y,z ودول بردوا متغيرات لكن جوه الدالة.. واضح إنهم مش زى بعض.

كل متغير تعرفه فى السكريت أو جوة دالة بيبقى ليه حاجة أسمها scope أو نطاق وليه life cycle دورة حياة هيعيش فى الرامة قد أيه !..

لما تعرف متغير فى أى مكان فى السكريت

```
x = 10
```

المتغير x اسمه global variable هيفضل جوة الرامة و تقدر تستخدمة وتستدعيه من أى مكان فى السكريت طول فترة تشغيل البرنامج.

طيب إيه الجديد ؟ لو المتغير ده جوة دالة function الوضع بيختلف .. المتغير ده أسمه local variable محدش بيشوفه بره الدالة..وقبل ما تعمل call للدالة المتغير ده مكنش موجود فى الوجود وبمجرد ما تعمل call للدالة والكود اللى جواها يتنفذ ويخلص المتغير ده بيتمسح من الرامة.



```
x=1
y=2

def xx():
    x=3
    y=4
    print('local x = %s'%x)
    print('local y = %s'%y)

print('global x = %s'%x)
print('global y = %s'%y)
xx()
print('again global x = %s'%x)
print('again global y = %s'%y)
```

```
global x = 1
global y = 2
local x = 3
local y = 4
again global x = 1
again global y = 2
>>>
```

Download script functions3.py from GitHub

- فى السكريت ده عملنا متغيرين global اسمهم x=1,y=2 . وجوة الدالة xx عملنا متغيرين local بنفس الاسم x=3,y=4 .
- خلى بالك المتغيرين x,y اللى بره global ملهمش علاقة بالمتغيرين x,y اللى جوة الدالة وأى تغيير هيحصل على الـ local مش هيأثر على الـ global.
- عشان تخلى المتغيرات الـ local اللى بتحمل نفس الاسم لمتغيرات global مرتبطة بيها بتعرف بتعرف المتغيرات الـ local اللى جوه الدالة بجملة global .

```
global var_name
```



x=1 y=2 def xx(): global x x=3 # x become global y=4 # y still local print('local x = %s'%x) print('local y = %s'%y) print('global x = %s'%x) print('global y = %s'%y) xx() print('again global x = %s'%x) print('again global y = %s'%y)	global x = 1 global y = 2 local x = 3 local y = 4 again global x = 3 again global y = 2 >>>

Download script functions4.py from GitHub

- لاحظ هنا خلينا المتغير x جوة الدالة بقى global والمتغير y زى ما هو local.
- لما خلينا قيمة المتغير x=3 ده غير القيمة فى المتغير الـ global .



البرمجة الكائنية OOP

البرمجة الكائنية أو Object Oriented Programming ده عبارة عن نمط وأسلوب برمجي بيعتمد على مفهوم الكائنات object والكلاسات (مش هترجمها فئات لا 😁) classes.

- الكلاس class : ده بناء برمجي فيه مجموعة من الداول وهنا تسمى methods ومتغيرات وتسمى attributes وهنشوف هنعمل بيهم أيه.

تعريف الكلاس فى لغة البايثون

```
class ClassName:
    variables...
    functions()..
```

- عشان تعرف الكلاس بتستخدم كلمة class وبعدها الاسم او الـ identifier. وفى البلوك بتاعها هيكون فيه مكونات الكلاس من دوال ومتغيرات.

قبل ما أعملك مثال على الكلاس لازم تعرف ان الدوال اللى فيها ليها مواصفات خاصة..

- الداول داخل الكلاس تسمى methods.

- المتغيرات داخل الكلاس تسمى class variables ودى أقدر أوصلها من أى دالة method جوة الكلاس أو من براها عن طريق object من الكلاس .



- يـمكـن تعريف دالة باسم __init__() وتسمى الـ class constructor ودى بتتنفذ بشكل تلقائى لما تعرف object من الكلاس وهقولك بعد شوية يعنى ايه object.

- كل الدوال اللى جوة الكلاس مكن نحط ليهم برامتر مكن تسميه أى اسم وفى الغالب اسمه self وده بيعبر عن الكلاس نفسها زى ما هتشوف.

ناخد مثال بقى ناخد مثال 😂.

```
class Animal:
    name=''
    pet=''
    prey=''
    enemy=''
    talent=''

    def __init__(self,name,pet,prey,enemy,talent):
        self.name=name
        self.pet=pet
        self.prey=prey
        self.enemy=enemy
        self.talent=talent

    def cv(self):
        cv='name:%s\nis pet:%s\nprey:%s\nenemy:%s\ntalent:%s'%(
        self.name,self.pet,self.prey,self.enemy,self.talent)
        print(cv)

    def on_danger(self):
        print('''when (%s) attacks (%s) it (%s)
              '''%(self.enemy,self.name,self.talent))
```

Download script class1.py from GitHub



- فى المثال ده عملنا كلا اسمها Animal.
- الكلاس Animal فيها 5 متغيرات name , pet , prey , enemy , talent ودى صفات للحيوان اللى بنعمله بالكلاس😁.
- المتغيرات دول اسمهم class variables ودول أقدر أوصلهم من أى دالة جوة الكلاس زى دالة __init__ و cv وon_danger.
- اول دالة فى بلوك الكلاس اسمها __init__ ودى الـ constructor بتاع الكلاس وبتتنفذ لما نعمل كائن من الكلاس.
- أستخدمنا الدالة __init__ عشان kuvt قيم لمتغيرات الكلاس class variables عن طريق تمرير برامترات فى الدالة __init__
- والمتغير self داخل كل دالة يشير للكلاس Animal نفسها.
- لو عاوز اوصل لأى متغير من class variables او أعمل call لـ class methods من داخل اى دالة فى الكلاس بكتب self قبل اسمها زى كده self. name أو self.cv() أو ممكن اكتب اسم الكلاس نفسها Animal.name
- شوفت جوة دالة __init__ لما جينا ندى قيم للمتغيرات name,pet,prey كان عندنا نوعين متغيرات بنفس الاسم.. النوع الأول متغيرات محلية خاصة بالدالة __init__ ودول البرامترات بتوع الدالة زى name مثلاً .. والنوع التانى المتغيرات التابعة للكلاس



واسمهم class variable وبوصلهم عن طريق self أو اسم الكلاس زي كده مثلاً self.name

- فيه عندنا دالتين cv دى بتطبع مواصفات الحيوان كأننا بنعمله cv ودالة on_danger بتطبع بيعمل ايه لما enemy يهاجمه.

اللى عملناه ده دلوقتى اسمه class او قالب لعمل داتا والمتغيرات بصفات وخصائص معينة أسمها objects او كائنات.

طيب ليه سماها objects ؟ قالك إن الكائن الحى بيميزه حاجتين .. الخصائص properties والسلوك behavior ولما يتكاثر.. الكائنات اللى بينتجها بترث inherit خصائص منه وهو حب يطبقلك نفس الأسلوب ده على البرمجة وهنشوف ازاى..

مثلاً الكلاس اللى عملناها Animal فيها متغيرات class variables بتحدد خصائص الحيوان زى name,enemy,pet وفيه دوال methods بتعبر عن سلوك الحيوان behavior ووضحتهالك أكتر بدالة on_danger مثلاً الحيوان هيعمل ايه وقت الخطر بناءاً على الخصائص أو المتغيرات بتاعته.

خلاص فهمت ليه اسمها برمجة كائنية ؟😁

يبقى اللى عملناه ده قالب او كلاس عاوزين نستخدمه ونعمل كائنات.. فهتعمل run للسكريبت ده وتتعامل مع الـ interactive interpreter .



```
>>>cat=Animal(name='Cat',pet=True,prey='Mouse',enemy='Dog',talent='Run Fast')
>>> cat.name
'Cat'
>>> cat.pet
True
>>> cat.cv()
name:Cat
is pet:True
prey:Mouse
enemy:Dog
talent:Run Fast
>>> cat.on_danger()
when (Dog) attacks (Cat) it (Run Fast)
```

- هنا عملنا object من الكلاس animal اسمه cat.
- مررنا الـ properties بتاعة cat كبرامترات فى الكلاس ودول هيروحوا للدالة __init__ أول الـ class constructor.
- الدالة __init__ هتاخد البرامترات دى وتعملها assignment للمتغيرات التابعة للكلاس أو class variables.
- من الكائن بتاع الكلاس cat أقدر أوصل لأى class variable وهنا يسمى attribute أو properties أو أى دالة زى cv و on_danger وهنا بنسميهم class methods وبوصلهم عن طريق الكائن اللى عملناه من الكلاس.



```
>>>giraffe=Animal('Giraffe', False, 'Vegetarian', 'Lion',
'Kick with legs')
>>> giraffe.cv()
name:Giraffe
is pet:False
prey:Vegetarian
enemy:Lion
talent:Kick with legs
>>> giraffe.on_danger()
when (Lion) attacks (Giraffe) it (Kick with legs)
```

- وهنا عملنا كائن تانى من الكلاس giraffe وعطيناله خصائص porperties وسلوك مختلف.

ممكن حد يسال هو فايدة البرمجة الكائنية إنى أقعد أعمل حيوانات وكلام فاضى😂؟

لا طبعاً ده مجرد مثال للتوضيح...وللعلم يعنى كل حاجة فى لغة البايثون تعتبر object تابعة لكلاس والكلاس دى فيها متغيرات ودوال أقدر أوصلها من خلال الـ object

```
>>> x = "Pink Python"
>>> type(x)
<class 'str'>
>>> dir(str)
[... 'find', 'format', 'format_map', 'index', 'isalnum',
'isalpha', 'isdecimal', 'isdigit', ...]
>>> x.upper()
'PINK PYTHON'
>>> x.split()
['Pink', 'Python']
```



- مثلاً لو عملت متغير نصى اسمه x .. فالمتغير ده يتبع للكلاس str ويحمل كل الدوال والمتغيرات اللى فيها واللى تقدر تعرفهم عن طريق الدالة dir(str) .
- وتقدر من خلال الـ object x تستخدم أى دالة من str زى الدوال اللى خدناهم فى موضوع الـ strings..

ده مثال تانى لكلاس اسمه Operations عملنا فيها دوال عشان تقوم بالعمليات الحسابية المعروفة جمع ووضرب وطرح وقسمة.

- ولاحظ أنى أستخدمت البرامتر me بدل من self عشان تعرف أنه ينفع يبقى أى اسم.

```
class Operation:
    num1=num2=0
    def __init__(me,num1,num2):
       me.num1=num1
       me.num2=num2
    def add(me):
        return me.num1+me.num2 #add num1 and num2
    def sub(me):
        return me.num1-me.num2 #subtract num2 from num1
    def mul(me):
        return me.num1*me.num2 #multiply num1 by mum2
    def div(me):
        return me.num1/me.num2 #divide num1 on num2
```



x=Operation(100,5) print(x.add()) print(x.sub()) print(x.mul()) print(x.div())	105 95 500 20.0
y=Operation(200,20) print(y.add()) print(y.sub()) print(y.mul()) print(y.div())	220 180 4000 10.0
x.num1=30 x.num2=5 print(x.add()) print(x.sub()) print(x.mul()) print(x.div())	35 25 150 6.0

Download script class2.py from GitHub

المثال ده ممكن يبينلك بعض فوائد استخدام البرمجة الكائنية والكلاسات وهو فى الأول التنظيم للكود على نمط وبناء محدد والتانى أعادة استخدام نفس الكود Code Re-usability بأننا عملت كلاس فيها 4 دوال وقدرنا نستخدم الدوال دى أكتر من مرة عن طريق الـ objects اللى بنعملها من الدوال.



الوراثة Inheritance

الوراثة تعتبر من أهم ميزات البرمجة الكائنية.. وقولنا إن الكائن ليه خصائص properties وهي المتغيرات وليه سلوك behavior وهي الدوال methods يقدر يرث كلاس أخرى عن طريق الـ Inheritance.

لما أخلي كلاس ترث كلاس تانيه.. الكلاس الأولى تسمى parent والكلاس التانيه تسمى child ولما يحصل وراثة كل الدوال والمتغيرات اللي في الـ parent هتروح للكلاس child ولما أعمل object من الكلاس child أقدر أستخدمهم بالإضافة إن الكلاس child بقدر أضيف فيه متغيراته ودواله الخاصه اللي مالهاش علاقة بالكلاس parent.

طريقة الوراثة :

```
class ParentClass:
    variables..
    functions()..

class ChildClass(ParentClass):
    variables..
    functions()..
```

عشان اخلي كلاس ترث التانيه.. يادوب بمرر الكلاس الـ parent للكلاس اللي عاوزه تورث child كبراميتر وبعدها أي object أعمله من Child يقدر يوصل لكل الـ properties والـ method في الـ parent.



ده مثال عشان يفهمك الوراثة وربنا يستر😁

``class Father:``` ``` father_name='Ahmed Pasha'``` ``` profession='Police Officer'``` ``` def job(me):print(me.profession)``` ```class Son(Father):``` ``` def __init__(me,name):``` ``` print(name,me.father_name)``` ```temo=Son('Tamer')``` ```temo.job()```	Tamer Ahmed Pasha Police Officer

Download script class3.py from GitHub

- هنا عملنا كلاس اسمها Father فيه متغيرين father_name, profession اسم ووظيفة الأب.. وفيها دالة job بتطبع المتغير بتاع الوظيفة profession.
- وبعدين عملنا كلاس تانى Son بترث الكلاس Father يعنى هتاخد كل المتغيرات والدوال اللى فى Father.
- تقدر دلوقتى تنسى إن فيه حاجة اسمها Father وتتخيل Son بقى شكلها كده

```
class Son:
    father_name='Ahmed Pasha'
    profession='Officer'
    def __init__(me,name):print(name,me.father_name)
    def job(me):print(me.profession)

temo=Son('Tamer')
temo.job()
```



- بعدين عملنا object من الكلاس Son ومررنا الاسم اللى هيروح لـ constructor وهو الدالة __init__ اللى هتطبع حاجتين name وهو البرامتر اللى باعتينه فى الكلاس Son و me. father_name وده اسم الأب الوارثينه من الكلاس Father.
- بعدين من الـ object اللى عملناه من الكلاس Son عملنا call للميثود job اللى هى أصلا كانت موجودة فى الكلاس Father وطبعت وظيفة الأب اللى أكيد هتروح أو راحت للأبن بعد الـ inheritance😂.

هناخد مثال تانى وهو تعديل على المثال الاول بتاع كلاس Animals.

```
class Animal:
    name=''
    pet=''
    prey=''
    enemy=''
    talent=''

    def __init__(self,name,pet,prey,enemy,talent):
        self.name=name
        self.pet=pet
        self.prey=prey
        self.enemy=enemy
        self.talent=talent

    def cv(self):
        cv='name:%s\nis pet:%s\nprey:%s\nenemy:%s\ntalent:%s'%(
        self.name,self.pet,self.prey,self.enemy,self.talent)
        print(cv)

    def on_danger(self):
        print('''when (%s) attacks (%s) it (%s)
              '''%(self.enemy,self.name,self.talent))


class Cats(Animal):
    family=''
    def cat_family(self):
        print(self.family)
```

Download script class4.py from GitHub



- هنا عملنا كلاس جديدة اسمها Cats بترث الكلاس Animals .
- وفى الكلاس الجديدة أضفنا متغر family و دالة cat_family.
- من خلال أى object للكلاس Cats أقدر اوصل لكل الدوال والمتغيرات اللى فى الـ parent و Animals والـ child وهو Cats.

```
>>> lion=Cats('Lion',False,'Small Animals','Big
Animals','Run Fast')
>>> lion.family='Big Cats'
>>> lion.cv()
name:Lion
is pet:False
prey:Small Animals
enemy:Big Animal
talent:Run Fast
>>> lion.on_danger()
when (Big Animals) attacks (Lion) it (Run Fast)
```

ده اللى ممكن تعرفه عن البرمجة الكائنية أو الكلاسات فى الكتاب ده او الإصدار الحالى.. ولو عاوز تكمل وتعرف اكتر فقدامك الموقع الرسمى classes.



جمل الشرط واتخاذ القرار Making Decision.

فى الأول كنا بنكتب الكود فى السكريبت ونعمل run فى الـ interpreter ينفذ الكود كله جملة جملة لحد ما يخلص.. بعدين عرفنا إن فيه حاجه اسمه function دى بتحط جواها شوية جمل برمجية وتقدر فى أى وقت تعملها call وخلى الـ interpreter يرجع عند السطر اللى فيه التعريف بتاع الدالة وينفذ الكود اللى فيها وبعدين يرجع يكمل البرنامج. كله ده بنتحكم فى سير البرنامج وطريقة تعامل الـ interpreter مع الكود اللى بنكتبه.

برده من طرق التحكم فى سير البرنامج control flow هى جمل الشرط condition.

جمل الشرط هى وسيلة لتنفيذ بلوك من الجمل البرمجية عندد تحقق شرط معين.

جملة If statement

أبسط صورة للشرط جملة if لوحدها.. بيكون شكلها كده

```
if condition :
   code...
```

```
if condition : one_line_code
```

- الحالة التانية طريقة كتابة جملة if والكود بتاعها فى سطر واحد..وبستخدم الشكل ده لو كان البلوك مثلاً جمله برمجية وحده فبعملها كده للتبسيط.
- الـ condition ده جملة برمجية إذا كانت لا تساوى False أو 0 او None البلوك بتاع if هيتنفذ.



- لو فاكر معايا فى جزء الـ operations وأتكلمنا عن عمليات المقارنة comparison operation العمليات المنطقية logical operations إن العمليات دى هى اللى بترجع True أو False .. ففى الغالب جملة الـ condition مع if هتلاقيها خليط من عمليات logical و relational زى ما هنشوف..
- وبالمناسبة هو ده الاستخدام الأساسى للـ relational والـ logical operations اللى أتكلمنا عنها فى أول الكتاب.. وهنشوف هنا أمثلة لبعض جمل الشرط

`>>> if True:print('yes')` `yes` `>>> if 1:print('yes')` `yes` `>>> if -1:print('yes')` `yes`	جملة الشرط ممكن تكون True أو أى رقم سالب أو موجب لا يساوى صفر وجميعهم هيحققوا الشرط
`>>> if False:print('yes')` `>>> if 0:print('yes')`	الـجملة لو كانت 0 أو False مش هيتحقق الشرط
`>>> x = 5` `>>> if x>4:print('yes')` `yes`	الشرط ممكن يكون relational operation x>4=True
`>>> x = 5` `>>> y = 'hello'` `>>> if x==5 and len(y)==4:` `        print('yes')` `>>> if x==5 and len(y)==5:` `        print('yes')` `yes`	الشرط ممكن يكون خليط بين الـ relational operations والـ logical operations عشان ننفذ حاجة فى حالة تحقق أكتر من شرط.



جملة else

فى حالة if لو الشرط تحقق هينفذ الكود بتاعها.. لو متحققش خلاص مفيش حاجة هتتنفذ...عشان كده عملوا جملة else.. لو الشرط بتاع if تحقق يتنفذ الكود بتاع if ولو متحققش يتنفذ الكود بتاع else.

```
if condition :
   code..
else:
   another_code..
```

- لاحظ إن جملة else بعدها : colon ومفيش condition لان الكود بتاعها هيتنفذ لو الشرط بتاع if متحققش بدون أى شروط

هناخد مثال على جملة if..else وهو سكريبت بيطلب أدخال رقم ..ويتعرف عليه إذا كان زوجي أو فردى.. وعشان تعرف الرقم زوجي ولا فردى بتقسمه على 2 وتشوف باقى القسمة لو يساوى 0 يبقى زوجي even لو يساوى 1 يبقى فردى odd.

- طبعاً عشان نجيب باقى القسمة بنستخدم المعامل % .

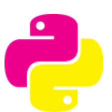
```
num = float(input('Enter a number : '))

if(num%2==0):print('%s is even.'%num)

else:print('%s is odd.'%num)
```

```
Enter a number : 5
5.0 is odd.
>>>
Enter a number : 10
5.0 is even.
>>>
```

Download script ifelse.py from GitHub



- لـما تعمل run وتشغل السكريت هيطلب منك تدخل رقم وبيشوفه زوجى ولا فردى لكن لو عاوز تجرب رقم تانى مش هينفع ولازم تعمل run من جديد عشان يبدأ الكود من اوله ودالة input تطلب منك تدخل الرقم..فانت محتاجه يكرر العملية.
- متخفش هنخلص موضوع الشرط ده وتمكن نلاقي حل قدام ولا حاجة 😂.

تخيل إنك عاوز تخليه يشوف شرط معين لو متحققش يشوف شرط تانى مختلف عن الأول

اللى بتفكر فيه بيخلينا نستخدم حاجة اسمها elif ودى في لغات تانيه اسمها else if

وبيكون شكل الشرط كالتالى..

```
if condition1 :
   code1..
elif condition2 :
   code2..
else:
   code3..
```

- لما الـ interpreter يجي ينفذ جملة الشرط اللى بالشكل ده دة بيشوف condition1 لو إتحقق هينفذ code1 بس ومش هينفذ لا code2..ولا code3.
- لو الشرط condition1 متحققش هينزل يشوف condition2 لو إتحقق هينفذ code2 بس.
- لو condition2 برده متحققش هيرح على code3..وينفذه من غير أى شروط.



- علـى فكـرة مش لازم جملة if يبقـى شابك فى ديلها else , زى الكـود اللـى تحتك ده فيه if و elif بس.. لو إتحقق condition1 هينفذ code1 ولو متحققش هيشوف condition2 لو أتحقق هينفذ code2 ولو متحققش يبقى خلاص مفيش حاجة هتتنفذ.

```
if condition1 :
    code1..
elif condition2 :
    code2..
```

عادى ممكن أجبلك جملتين if تحت بعضيهم مش if وتحتها elif بس فى الحالة دة كل شرط هيبقى منفصل عن التانى .

x = y = 1 if x>0:print('x') if y>0:print('y')	x y	x = y = 1 if x>0:print('x') elif y>0:print('y')	x



آخر حاجة هنعمل مثال شامل لجملة if ..elif..else عشان يلخص الموضوع كله.

- المثال بيطلب من المستخدم يدخل نسبته المئوية ومنها يـحسب له التقدير والـ GPA.

<pre>x = int(input("Enter your percentage : ")) if(x >= 85): print("Excellent") if(x>=95):print('A+') elif(x>=90):print('A') else:print('A-') elif(x >= 75): print("Very Good") if(x>=80):print('B+') else:print('B') elif(x >= 50): print("Good") if(x>=70):print('B-') elif(x>=60):print('C') else:print('D') else: print("Failed")</pre>	<pre>Enter your percentage : 91 Excellent A >>> Enter your percentage : 80 Very Good B+ >>> Enter your percentage : 60 Good C >>> Enter your percentage : 40 Failed >>></pre>

Download script condition.py from GitHub

- طبعاً عشان تحسب النسبة المئوية لأكتر من شخص لازم تعمل run للسكريت عشان يبدأ من أول سطر ويخلي دالة input تطلب منك تدخل النسبة المئوية من جديد ودى تعتبر مشكلة فى البرنامج.

- ومشكلة تانية إن الـ input اللى بتكتبه لازم يكون أرقام عشان بنحوله لمتغير عددى بدالة int ولو المستخدم كتب حروف هيحصل خطأ والبرنامج هيوقف.



جمل التكرار Loop

من شوية خليتك تعيش مشكلة معايا ومش متأكد إذا كنت حسيت بيها أصلاً 😁 بس شوفنا لما كنت بتحسب النسبة المئوية لأكتر من شخص كان لازم تعمل run للبرنامج من جديد عشان يبدأ الكود من الاول ويطلب منك تدخل النسبة المئوية.. فالموضوع رخم بعض الشيء ومش عملي..

ولو فرضنا مثلاً إنك عاوز تطبع على الشاشة الأرقام من 1 إلى 100 هل هتقعد تكتب print() 100 مرة 🤔؟ .. بالتأكيد لأ عشان كده عملوا جمل التكرار .

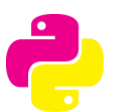
وهي جمل بتخلي الـ interpreter يكرر تنفيذ بلوك من الكود عدد من المرات بيحدده الشرط بتاع جملة التكرار.

جملة while loop

جملة while بتنفذ الكود اللى فى البلوك بتاعها طول ما الـ condition بتاعها متحقق وبتتكتب بالشكل ده ..

```
while condition:
    code..
```

```
while condition:single_line_code...
```



- طول ما الـ condition ده قيمته True البلوك بتاع جملة while كل ما يخلص تنفيذه هيتنفذ تانى وهكذا لحد ما الـ condition ميبقاش True فنخرج من التكرار.
- لو خليت الـ condition يساوى True ومفيش حاجة تغيره..التكرار هيفضل لما لا نهاية (يعنى لحد ما تقفل الـ interpreter) فمتجربش... أو جرب الكود ده فى الخباثة من غير ما تقولى 😂

```
while True:print("loop")
```

- هتلاقية بيطبع جملة loop كتير ومش بيتوقف وممكن يخلى الجهاز يهنج.
- يبقى لازم الشرط condition يكون True عشان التكرار يبدأ ويكون محكوم بحاجة تخليه False بعد عدد مرات تكرار معينة بحددها أنا عشان يوقف.

مثلاً عاوز أعمل سكريبت يطبع الأرقام من 0 إلى 5 .. فهيكون بالشكل ده.

i = 0 while i < 6 : print(i) i=i+1 print('Done') print('i =',i)	0 1 2 3 4 5 Done i = 6	هنا عملنا متغير i=0 بعدين الشرط بتاع while هو إن قيمة i تكون أقل من 6. لو تحقق الشرط هيطبع المتغير i ويزود عليه واحد i=i+1 وبعدين ينفذ البلوك تانى يطبع ويزود. التكرار هيستمر لحد ما i توصل لـ 6 فالشرط هيبقى 6<6 = False .. فيخرج من التكرار ويطبع جملة Done وأخر قيمة وصل ليها i وهو 6

Download script while.py from GitHub



الأمر break

بنستخدم الأمر break للخروج من جملة التكرار فى أى وقت بغض النظر عن حالة الـ condition بتاع التكرار.

- مثلاً فى الكود اللى تحتك الشرط i<=10 يعنى لو سبته عادى هيطبع الأرقام من 0 إلى 10 .. لو عاوزين نوقف التكرار لما i=5 ومنستناش لحد ما تبقى أكبر من 1, هنعمل جملة شرط تتحقق لما i==5 فيعمل break.

i = 0 while i <= 10 : print(i) if(i==5):break i=i+1 print('Done') print('i =',i)	0 1 2 3 4 5 Done i = 5

Download script while_break.py from GitHub

- خلى بالك جداً لما عمل break كانت قيمة i =5 ومكملش البلوك بتاع while يعنى منفذش السطر الأخير فيه i=i+1 وكانت قيمة i النهائية 5.

ومكن أستخدم infinite loop يعنى أخلى condition قيمته True والمفروض التكرار يستمر على طول بس أوقفه عند حد معين باستخدام break داخل شرط.



<pre>i = 0 while True : print(i) if(i==5):break i=i+1 print('Done') print('i =',i)</pre>	<pre>0 1 2 3 4 5 Done i = 5</pre>

الأمر continue

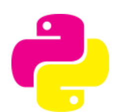
بيخلي الـ interpreter يرجع لبداية بلوك جملة while حتى لو مكنتش خلص الجمل في البلوك ويبدأ لوب من جديد.

المثال ده هيوضحلك اكتر.. عاوزين نطبع الأرقام الفرديه من 0 إلى 10.

<pre>i = 0 while i < 10 : i=i+1 if i%2==0:continue print(i)</pre>	<pre>1 3 5 7 9</pre>

Download script while_continue.py from GitHub

- الكود ده مختلف شوية عن اللي قبله..هنا في أول سطر في البلوك بنعمل increment للمتغير i وهقولك ليه.
- وبعدين جملة الشرط اللي بتتحقق لو باقي قسمة i على 2 يساوي 0 وده في حالة الأرقام الزوجية وساعتها هيعمل continue يعني يخلي الـ loop دي ويبدأ loop جديدة من غير ما يكمل الكود اللي بعد الأمر continue



وهو جملة print اللى بتطبع قيمة المتغير.

- خلى بالك إن الامر break بينهى التكرار تماماً ويطلع من while لكن continue بيطلع من الـ loop قبل ما تخلص ويدخل على اللى بعدها.

واحد هيسالنى ليه عملت الـ increment على المتغير i فى أول البلوك مش فى اخره ؟🤔.

```
i = 0
while i <= 10 :
   print(i)
   if i%2==0:continue
   i=i+1
```

- مش هقولك ليه بس ممكن تجرب الكود ده وهتلاقيه بيطبع 0 إلى ما لا نهاية..
- وده عشان فى البداية i = 0 فهيطبع 0 ويدخل على الشرط.. هل 0==0%0 نعم فهيحقق الشرط ويعمل continue ومش هيكمل باقى البلوك ولا يوصل لـ i=i+1 يعنى i لسه 0 زى ما هى وهيبدأ لوب جديده.
- فى اللوب الجديده هيطبع i اللى لسه 0 والشرط هيتحقق فهيعمل continue قبل ما يزود وهيستمر كده لمالا نهاية.
- عشان كان لزاماً علينا إننا نخلى الـ increment فى أول البلوك عشان يلحق يزود رقم على i قبل جملة continue.



جملة while ... else

اللى عملوا لغة البايثون حبوا يضيفوا حاجة جديدة فى جملة while عن اللى بنشوفه فى اللغات التانيه هو حطوا جملة else بتاعة if فاكرينها طبعاً .. جابوها هنا برده عشان الشرط condition بتاع جملة while ميتحققش والتكرار يخلص ينفذ البلوك بتاع جملة else .

شكل جملة while..else بيكون كده

```
while condition:
   statements..
else:
   another statements.
```

- لما condition يتحول الى False أو لو كان False هينفذ الكود بتاع جملة else.

<pre>i = 10 while i < 6 : print(i) i=i+1 else: print('i =',i) print('condtion == False')</pre>	<pre>i = 10 condition == False</pre>

- مثلاً هنا.. من الاول الـ condition 10<6 = False فمدخلش فى جملة التكرار ونفذ جملة else على طول.



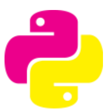
<pre>i = 0 while i < 6 : print(i) i=i+1 else: print('i =',i) print('condtion == False')</pre>	<pre>0 1 2 3 4 5 i = 6 condition == False</pre>

Download script while_else.py from GitHub

- وهنا شرط التكرار تحقق لحد ما i بقيت 6 وبعديها طلع من جملة while ونفذ جملة else.
- و خلي بالك جملة else بتتنفذ في حالة إن الشرط بتاع while كان False أو تحول إلي False.
- أما لو عملت break في جملة while وخرج منها.. مش هينفذ جملة else.

<pre>i = 0 while i < 6 : print(i) if(i==3):break i=i+1 else: print('i =',i) print('condtion == False')</pre>	<pre>0 1 2 3</pre>

Download script while_else.py from GitHub



قبل ما نـخلـص كـلام عن while , قولتلك هنجمع حل لمشكلة السكريت ده condition

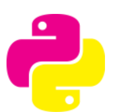
اللى بيحسب التقدير من النسبة المئوية والمشكلة اللى كانت فيه إن بيطلب منك تدخل

النسبة المؤية مرة وحدة وبعد كده لازم تعمل run تانى عشان يبدأ من اول السطر.

- الحل إننا نحط الكود فى بلوك لجملة while وخليه يكرر نفسه وبكده مش هنحتاج نعمل run للسكريت كل مرة.

<pre>while(True): x = int(input("Percentage : ")) if (x==-1):break if(x >= 85): print("Excellent") if(x>=95):print('A+') elif(x>=90):print('A') else:print('A-') elif(x >= 75): print("Very Good") if(x>=80):print('B+') else:print('B') elif(x >= 50): print("Good") if(x>=70):print('B-') elif(x>=60):print('C') else:print('D') else: print("Failed") print("==================")</pre>	<pre>Percentage : 99 Excellent A+ ================== Percentage : 58 Good D ================== Percentage : 85 Excellent A- ================== Percentage : 60 Good C ================== Percentage : -1 >>></pre>

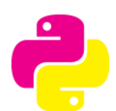
Download script while_loop.py from GitHub



- حطينا الكود فى infinite loop الشرط بتاعها True الحل الوحيد إن التكرار ده يخلص هو تحقق الشرط بتاع if (x==-1):break بأن اليوزر يدخل رقم -1 وفى الحالة دى يعمل break وخرج من البرنامج.
- من الحاجات اللى ممكن تكون لاحظتها أو لسه ملاحظتهاش إن دالة int() اللى بتحول النص بتاع دالة input لمتغير عددى لازم النص ده يكون أرقام.
- يعنى لو غلطت ودخلت حروف هيحصل error لأنه مينفعش يحول النص ده إلى أرقام والبرنامج هيوقف.

```
Percentage : s20
x = int(input("Percentage : "))
ValueError: invalid literal for int() with base 10: 's20'
```

ده بيخلق عندنا مشكلة تانيه هنشوف حلها قدام إزاى لما يحصل خطأ غير متوقع من المستخدم نقدر نعالجه من غير ما البرنامج يوقف..هنتكلم عليها فى الفصل بتاع الأخطاء Exceptions.



جملة for iteration

جملة for في لغة البايثون بتختلف كتير عن الموجوده لغات c/c++..و الحاجة الوحيدة اللى بتعملها for في البايثون هو الـ iteration أو التكرار على مجموعة من العناصر.

- طريقة استخدامها بيكون بالشكل ده

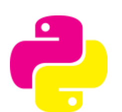
```
for item in sequence:
    code..
```

- شغل جملة for بيكون على data sequence معين زى list أو dictionary أو tuple أو حتى string.. وجملة for كل إللى بتعمله إنها بتفصص الـ sequence ده عنصر عنصر في كل loop والأمثلة هتوضحلك أكتر..

`numbers=[1,2,3,4,5]` `for number in numbers:` `    print(number)`	1 2 3 4 5

- في المثال ده عندنا list اسمها numbers .. عملنا iteration على الـ items بتوعها.
- بحيث إن كل مرة في التكرار يطلع عنصر من numbers ويحطه في المتغيرnumber.

`x='HELLO'` `for letter in x:` `    print(letter)`	H E L L O



- فى المثال ده الـ sequence اللى عملنا عليه تكرار هو نص string يعنى مجموعة من الحروف فى كلمة HELLO.
- أكيد أنت فاهم إن المتغير letter أو number فى المثال ده و اللى فوق هما متغير التكرار iteration variable وينفع نسميه أى اسم طبعاً.

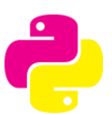
استخدام for .. in range

لو كنت تعاملت مع أى لغة برمجة تاني هتقولى مش دى for اللى انا أعرفها 😂 أنا عاوز for اللى فيها بداية init و condition يخليها توقف عند حد معين و increment اللى بتحدد مقدار زيادة متغير الـ index بتاعها لحد ما يخلى الـ condition يبقى False و يوقف الـ for .

الكلام ده عملته لغة البايثون بشكل مختلف فى دالة range..مش هشرحلك for in range قبل ما أفهمك دالة range بتعمل ايه.

دالة range بتولد متسلسلة من الأرقام ببداية ونهاية ومعدل زيادة أنا يحدده.

`range(start,stop,step)`	دى الصورة الكاملة لبرامترات الدالة
`range(start,stop)`	كده بيعتبر step = 1
`range(stop)`	كده بيعتبر step = 1 والبداية start =0



دالة range ممكن تاخد 3 برامترات البداية start والنهاية stop ومقدار الزيادةstep.. عشان تولد متسلسلة من الأرقام تبدأ من رقم start بتزيد بمقدار step وأكبر رقم فيها بيكون أصغر من stop.

`>>> list(range(1,10,1))` `[1, 2, 3, 4, 5, 6, 7, 8, 9]` `>>> list(range(0,10,2))` `[0, 2, 4, 6, 8]`	`>>> list(range(2,8,1))` `[2, 3, 4, 5, 6, 7]` `>>> list(range(3,33,3))` `[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]`

بنستخدم دالة list عشان نحول المتسلسلة إلى list ونشوف الأرقام اللى فيها.

خلى بالك إن أخر رقم فى الليسته بيبقى أقل من قيمة الـ stop مش بيوصل له.

`>>> list(range(1,5))` `[1, 2, 3, 4]` `>>> list(range(1,5,1))` `[1, 2, 3, 4]`	`>>> list(range(5))` `[0, 1, 2, 3, 4]` `>>> list(range(0,5,1))` `[0, 1, 2, 3, 4]`

هنا لما مررنا برامترين بس للدالة.. أعتبرتهم start و stop وخلت الـ step = 1

لو مررنا برامتر واحد أعتبرته stop وخلت البداية من صفر مش 1 و الـ step = 1

`>>> list(range(1,5,1))` `[1, 2, 3, 4]` `>>> list(range(5,1,-1))` `[5, 4, 3, 2]`	`>>> list(range(4,0,-1))` `[4, 3, 2, 1]` `>>> list(range(20,-2,-2))` `[20, 18, 16, 14, 12, 10, 8, 6, 4, 2, 0]`



- لو عاوز المتسلسلة تكون تنازلية..بتخلى الـ start أكبر من الـ stop والـ step بالسالب...وبرده أصغر رقم فى المتسلسلة هيكون أكبر من الـ stop.

طيب .. مش احنا قولنا إن for بتعمل iteration على أى sequence ؟ حصل وقولنا إن range بتطلع sequence فبالتالى نقدر نستخدم for مع range ونعد بتسلسل معين ليه بداية ونهاية ومعدل زيادة أنا بحدده كأننا بنستخدم جملة for فى لغة الـc/c++ واللغات التانيه.

يبقى جملة for شكلها هيبقى زى كده.

```
for index in range(start,stop,step):
    code..
```

دلوقتى عاوز أعمل برنامج بيعد من 0 إلى 10 بمعدل زيادة 2 .. وعملتلك جزء من الكود بلغة الـ C++ عشان تفهم الفرق.

`// c++ for loop` `for(int i = 0 ; i < 12 ; i=i+2)` `   cout<<i<<endl;`	`# python for loop` `for i in range(0,12,2):` `   print(i)`

```
0
2
4
6
8
10
```



- خلينا فى كود البايثون .. كل اللى عمله إنه عمل ليسته من الـ range(0,12,2).

الليسته دى شكلها كده

```
>>> list(range(0,12,2))
[0, 2, 4, 6, 8, 10]
```

وقعد يعمل iteration فيها وطلع العناصر واحد واحد وطبعها.

- وطبعاً جملة for تقدر تستخدم فيها الأمر break اللى بيخرج من الـ loop وجملة continue اللى بترجع لبداية البلوك قبل ما تخلص الـ loop والمثال ده بيوضحلك..

`for i in range(22):` `    if(i==10):break` `    if(i%2==0):continue` `    print(i)`	1 3 5 7 9

Download script for_range.py from GitHub

فى الكود ده دالة range خدت برامتر واحد stop وبالتالى هتعمل متسلسلة بدايتها 0 و بتزيد بمعدل 1 يعنى المفروض for تعد من 0 لحد 20.

- الشرط الأول لما i يبقى 10 أو يوصل للعنصر اللى قيمته 10 فى الـsequence ده يعمل break وبكده ضمنا إن التكرار مش هيستمر لحد الرقم 20 وهيوقف عند 10.
- الشرط التانى إن يقسم الرقم على 2 لو باقى القسمة % ساوى 0 يعنى الرقم زوجى فى الحالة دى يعمل continue ويرجع ينفذ loop جديدة قبل ما يخلص باقى الكود اللى هو بيطبع قيمة i وبالتالى الأرقام الزوجية مش هتتطبع.



وده مقارنة بين المثالين اللى بيطبعوا الأرقام الزوجية من 0 إلى 10 باستخدام while و for كمقارنة بينهم.

`i = -1` `while i <= 10 :` `    i=i+1` `    if i%2:continue` `    print(i)`	`for i in range(11):` `    if i%2:continue` `    print(i)`

فى حالة while خليت i=-1 عشان بيعمل increment قبل ما يطبع الرقم .. فلو كان i=0 هيزودها فتبقى 1 ومش هيطبع 0 كرقم زوجى.

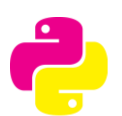
فى حالة الـ for stop بتاع دالة range 11 عشان زى ما انت فاكر أخر رقم فى المتسلسلة اللى بيعملها بيكون أصغر من الـ stop اللى هو رقم 10.

جملة for .. else

زى جملة while بالزبط تقدر تستخدم else مع for عشان تنفذلك حاجة معينة بعد إنتهاء الـ iteration.

`for i in range(4):` `    print(i)` `else:` `    print('done..')`	`0` `1` `2` `3` `done..`

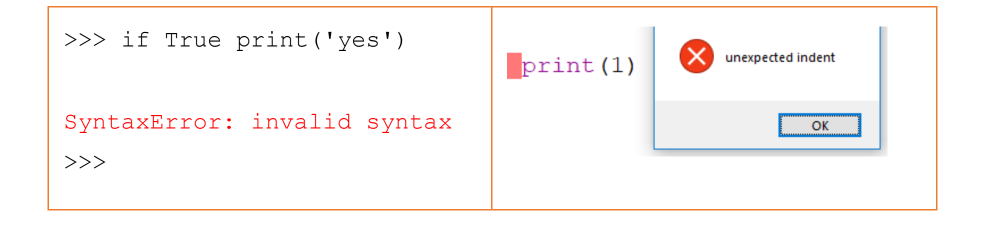
Download script for_else.py from GitHub



الأخطاء والاستثناءات Errors and Exceptions

الأخطاء Syntax Errors

لغة البايثون فيها نوعين من الأخطاء.. الأول متعلق بقواعد اللغة نفسها مثلاً فى كتابة تعريف الدالة أو أى نوع من البلوكات فتنسى الـ colon : ده اسمه SyntaxError أو لو غلطت فى محاذاة أسطر البلوك أو سطر أديته مسافة زياده بالغلط اسمه IndentationError. النوع ده يسمى خطأ قاتل fatal error.. مفيهوش تسامح ولازم تصححه ولو كان السكريت 100 سطر وسطر واحد فيه الخطأ من النوع ده السكريت مش هيشتغل أصلاً.



```
>>> if True print('yes')

SyntaxError: invalid syntax
>>>
```

```
print(1)
```


الإستثناءات Exceptions

النوع التانى من الأخطاء أو تسمى exceptions.. فى الحالة دى القواعد البرمجية syntax سليمة ولو عملت تشغيل للكود.. الـ interpreter هيبدأ ينفذ الكود ومفيش مشاكل لحد ما يقابله الخطأ أثناء التشغيل runtime تخلى البرنامج يوقف.. وأنواعه كتير.. مثلاً إنك تعمل call لدالة أو متغير قبل ما تعرفه فهيطلعلك خطأ NameError ولو عاوز تجمع رقم فى متغير عددى مع رقم فى متغير نصى زى كده 5+'5' وده عدى علينا قبل كده



هيتنج TypeError أو لو حاولت تقسم على صفر وده ليس له معنى ياضياً فهيطلعلك ZeroDivisionError .. وغيرهم كتير.

وهنا فى الوثيقة بتاعة اللغة جايبلك كل الـ Exceptions اللى ممكن تظهرلك فى لغة البايثون bltin-exceptions .. مش لازم تعرفهم كلهم أنت هتشوفهم بنفسك 😁.

معالجة الأخطاء Handling Exception

أياً كان نوع الأخطاء ولو كانت exceptions برده هتخلى السكريت يوقفك وميكملش.

عشان كده عملوا جملة try اللى شكلها بيكون كده.

```
try:
  something_to_do
except:
    something_else.
```

- اللى بيحصل إن الـ interpreter بيحاول ينفذ البلوك بتاع جملة try ولو حصل Exception هينفذ البلوك بتاع جملة except.

هناخد مثال على طول على جملة try..except وهو سكريت الأرقام الزوجية والفرديه. المستخدم يدخل رقم والسكريت يقراه عن طريق دالة input ويحدد إذا كان زوجي أو فردى زى ما أخدناه قبل كده.



```
while True:
  x = int(input('Enter A Number : '))
  if x%2==0:print('Even')
  else:print('Odd')
```

```
Enter A Number : 1
Odd
Enter A Number : 2
Even
Enter A Number : f
ValueError: invalid literal
for int() with base 10: 'f'
```

- فى المثال ده دالة int بتاخد الرقم اللى بيكتبه المستخدم من دالة input اللى بترجعه على شكل متغير نصى وتحوله دالة int لمتغير عددى فى x.
- وبعدين لو كان باقى قسمته على 2 يساوى صفر يبقى زوجى even غير كده odd.
- المشكلة لو المستخدم دخل حروف مش أرقام.. دالة int لما تاخد الحروم من دالة input وتيجى تحولهم هيطلع خطأ ValueError هيوقف البرنامج.

الحل إننا نحط الكود الخطير ده فى جملة try..except.

```
while True:
    try:
        x = int(input('Enter A Number : '))
    except:
        print('incorrect number')
        continue
    if x%2==0:
        print('Even')
    else:
        print('Odd')
```

```
Enter A Number : 1
Odd
Enter A Number : 2
Even
Enter A Number : f
incorrect number
Enter A Number : 3
Odd
Enter A Number : d
incorrect number
```

Download script try1.py from GitHub



- هنا الـ interpreter هينفذ بلوك جملة try..ولو المستخدم دخل حرف مش رقم وحصل Exception هينفذ البلوك بتاع except اللى يطبع كلمة incrorrect number ويعمل continue يعنى يرجع من اول البلوك بتاع while ويطلب رقم من جديد.

ومـمكن تستخدم جملة try..except عشان تنفذ كود بناءاً على نوع الـ Exception اللى هيطلع وهيكون شكلها كده.

```
try:
  something_to_do
except <error_name1>:
    something_else.
except <error_name2>:
    something_else.
```

فى الحالة دى لازم تكون عارف اسماء الـ Exceptions المتوقع إنها تظهر معاك.. و ده مثال لسكريت يطلب من المستخدم إنه يدخل رقمين num1 و num2 ويقسم الأول على التانى.

- فى الحالة دى الأخطاء المتوقعة تحصل إن المستخدم يدخل حروف فلما اجى أحوله لرقم بدالة int هيظهر ValueError.
- وفيه خطأ تانى ممكن المستخدم يدخل الرقم num2 صفر والقسمة على صفر بترجع خطأ ZeroDivisionError .



<pre>while True: try: num1 = int(input('Enter A Num1 : ')) num2 = int(input('Enter A Num1 : ')) result=num1/num2 print(result) except ZeroDivisionError: print('Cant divide on zero') except ValueError: print('Incorrect Number')</pre>	<pre>Enter A Num1 : 6 Enter A Num1 : 3 2.0 Enter A Num1 : 5 Enter A Num1 : 0 Cant divide on zero Enter A Num1 : f Incorrect Number Enter A Num1 : 1 Enter A Num1 : g Incorrect Number</pre>

Download script try2.py from GitHub

أخر حاجة هتكلم عنها فى الموضوع ده هو جملة finally اللى بتتضاف لجملة try..except

```
try:
  something_to_do.
except:
  something_else.
finally:
  do_whatever.
```

قولنا إن المفسر بيحوال ينفذ بلوك جملة try ولو ظهر خطأ بيروح لجملة except.

فأنا لو عاوز أنفذ كود فى الحالتين بقى سواء كان فيه خطأ أو مفيش هينفذ فبستخدم

جملة finally.. وهديك مثال يوريك ممكن تستخدمها فى أيه.



marks={'ahmed':20,'ali':19, 'mahmoud':10,'noor':18} name=input('Enter your name : ') try:mark=marks[name] except:mark='user not found' finally:print(mark)	Enter your name : ahmed 20 Enter your name : salah user not found

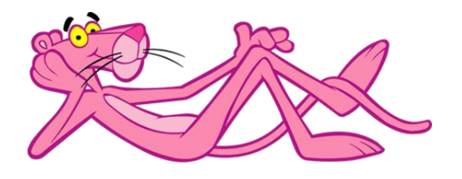
Download script try3.py from GitHub

- فى المثال ده عملنا dictionary فيه اسماء الطلاب ودرجاتهم ومطلوب من المستخدم يدخل اسمه عشان يستخدمه كـ key فى الـ dict ويجيب له الـ value المقابلة وهى الدرجة.
- لو الاسم موجود هيرجع الـ value المقابلة ليه فى المتغير mark ولو مش موجود هيحصل exception لأنك بتحاول تطلع قيمة لـ key مش موجود أصلاً فهينفذ جملة except اللى بتحط فى المتغير mark النص user not found.
- فى الحالتين حصل exception أو محصلش كود جملة finally هيتنفذ وهتطبع هتطبع قيمة المتغير mark أياً كان.

وده كل اللى أقدر أقولهولك وهيفيدك فى موضوع الـ exception.. وطبعاً تقدر تشوف الموضوع على توسع أكتر فى الـ documentation بتاعة البايثون من هنا errors



الفصل 3 – المكتبات فى لغة البايثون Libraries



الفهرس



الموديولات Modules

لو بتلاحظ لما تيجى تعمل run للسكريت بتاعك على الـ IDLE بتدوس على قائمة run >> run module يبقى الموديول ده هو السكريت أو الكود بتاعك اللى بيكون فيه مجموعة من المتغيرات variables الدوال functions أو الكلاسات classes.

لغة البايثون بتتيح ليك إنك تقسم الكود بتاعك على شكل موديولز أو سكريتات عشان التنظيم وعشان ميبقاش الكود كله فى سكريت واحد ويبقى طويل.. وكل سكريت من دول تقدر تعمله استدعاء داخل السكريت الأساسى بتاعك عن طريق جملة import.

يعنى الموديولز دى زى ملفات class files فى لغة الجافا او الـ libraries المعروفة فى أغلب اللغات.

استدعاء موديول importing modules

طريقة استدعاء object (دالة , متغير,class) من موديول بيكون بالشكل ده وبالأمثلة هنعرف الفرق بينهم..

```
import module_name
from module_name import object
from module_name import *
```



هنعمل سكريت جديد أو موديول جديد وهنعمل فيه دالة اسمه fact ودى تحسب الـ factorial أو مضروب العدد , لو مش فاكر يعنى أيه مضروب العدد هو إنك بتجيب العدد وتضربه فى نفسه وفى كل الأرقام اللى أصغر زى 4 مضروبه = 4 * 3 * 2 * 1 = 24

هتسمى الموديول factorial.py وتحفظه وتعمله Run..

`def fact(n):` `    num = 1` `    while n >= 1:` `        num = num * n` `        n = n - 1` `    return num`	`>>> fact(5)` `120` `>>> fact(4)` `24` `>>>`

- لما تشغله مفيش حاجة هتظهر لاننا عرفنا الدالة بس معملنلهاش استدعاء call .
- اسم الدالة fact بنمرر ليها العدد اللى عازين نجيب مضروبة كبرامتر اسمه n واستخدمنا فيها جملة التكرار while عشان كل مرة تنقص 1 من العدد وتضربه فى المتغير num وفالنهاية ترجع المتغير num.
- لما تعمل run فى الـ interactive interpreter أبقى أستدعى الدالة ومرر ليها الرقم اللى عاوز تجيب المضروب بتاعه بالشكل ده

```
>>> fact(5)
120
```

دلوقتى هنعتبر الموديول factorial ده مكتبه وعاوزين نستخدمه فى سكريتات تانيه لينا. هنعمل نعمل سكريت جديد ونستدعى جواه الموديول اللى اسمه factorial ونستخدم الدالة fact اللى بتحسب المضروب.



```
import factorial
x = factorial.fact(4)
print(x)
```

```
>>> 24
```

- فى السطر الأول عملنا import للموديول factorial وخلى بالك من نقطتين..

الأولى إنك مكتبتش factorial.py كتبت اسمه بس والنقطة التانية لازم تاخد بالك من مكان الموديول factorial.py بالنسبة للموديول اللى احنا مشغلينه دلوقتى.

عشان لما تعمله import الـ interpreter يعرف يجمعه وينفذ الكود اللى جواه.

الأماكن اللى تحط فيها الموديول عشان تقدر تعمله import

1. فى نفس المجلد اللى فيه الموديول اللى هتعمله run
2. فى المجلد الرئيسى بتاع لغة البايثون .
3. فى المجلد Lib موجود فى جودة مجلد البايثون.

- المهم تانى سطر فى الكود عملنا call للدالة اللى اسمها fact فى الملف factorial بالشكل ده factorial.fact()

- وطبعاً الدالة بترجع المضروب فخدناه فى متغير x وعملناله print .

لاحظ إنك لو مكتبتش اسم الموديول وحاولت تستدعى الدالة fact على طول مش هيلاقيها وهيطلع NameError .

```
x = fact(5)
NameError: name 'fact' is not defined
```

وعشان نحل المشكلة دى تقدر تستدعى الدالة fact نفسها من الموديول بالشكل ده.



```
from factorial import fact
x = fact(4)
print(x)
```

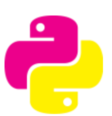
خلي بالك الفرق بين import module_name و import object from module_name

- إن الطريقة الاول بتستدعى كل الداول والمتغيرات والكلاسات إن وجدت اللى فى الموديول ولكنى بقدر أوصلها عن طريق اسم الموديول module_name.object
- أما الطريقة التانية بتستدعى الـ object اللى بعمله import بس من الموديول و بقدر أوصلها عن طريق اسمها مباشرة object_name بس لو فيه دوال تانى فى الموديول مقدرش اوصلها لأنى كده معملتلهاش import.
- عشان تفهم أكتر..ارجع للموديول factorial عشان هنضيف فيه دالة تانيه نسميها fact2 تحسب المضروب المرة دى بس بجملة for

```
def fact(n):
    num = 1
    while n >= 1:
        num = num * n
        n = n - 1
    return num
def fact2(n):
    num = 1
    for i in range(1,n+1):
        num = num * i
    return num
```

```
>>> fact(5)
120
>>> fact2(5)
120
>>>
```

Download script factorial.py from GitHub



- دلوقتى بقى عندنا دالتين هما الأتنين بيعملوا نفس الوظيفة يحسبوا المضروب بس وحدة جملة while والتانية جملة for .
- ياريت تكون لسه فاكر شرح دالة range فى for وعارف ليه البرامتر التانى بتاعها n+1 ..

المهم هنرجع للسكريت التانى اللى بنعمل منه import للمكتبة اللى بنعملها بتاعة factorial وهنعدل عليه بالشكل ده..

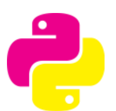
`from factorial import fact` `x1=fact(5)` `x2=fact2(5)` `print(x1)` `print(x2)`	`NameError: name 'fact2' is not defined`

- تلاحظ إنه عملك error وقالك إنه مش عارف fact2 دى تبقى مين وطبعاً حقه لأنك فوق عملت import للدالة fact بس من الموديول كان لازم تعمل import fact,fact2.
- او ممكن تعمل import للموديول كله module import ولما تعمل call للدالة تكتب اسم الموديول قبلها

`import factorial` `x1= factorial.fact(5)` `x2= factorial.fact2(5)` `print(x1)` `print(x2)`	`120` `120` `>>`

- لو مدايقك طول اسم الموديول factorial ممكن تستدعيه بأى باسم قصير

import module as name



```
import factorial as f
x1= f.fact(5)
x2= f.fact2(5)
print(x1)
print(x2)
```

```
120
120
>>
```

- أو تقدر تعمل import لكل الدوال اللى فيه بالشكل ده from module import *

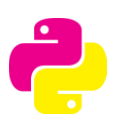
بدل ما تكتب اسم كل دالة بعد كلمة import.

```
import factorial import *
x1= fact(5)
x2= fact2(5)
print(x1)
print(x2)
```

```
120
120
>>
```

يبقى دى طرق استخدام الموديلوز أو المكتبات سواء أنت عملتها بنفسك أو كانت جاهزة تبع اللغة أو حملتها من على الأنترنت.

موديلوز تبع اللغة! 🤔 أيوة لغة البايثون بيجى معاها مجموعة كبيرة من المكتبات تقدر تستخدمها فى تطبيقاتك المختلفة وهنتكلم عنها الجزء اللى جاى..



مكتبة بايثون الأساسية The Python Standard Library

سهولة أى لغة برمجة بيعتمد بشكل كبير على مدى ثراء المكتبة الأساسية standard Library التابعة لها.

لغة البايثون عبارة عن مجموعة من القواعد البرمجية اللى تخليها مختلفة عن أى لغة تانيه بالإضافة لمجموعة من الدوال functions وأنواع البيانات data types والمكتبات أو موديلوز modules بيسهلوا عليك استخدام اللغة وبيساعدوك تنفذ كل التاسكات البرمجية اللى بتحتاجها كمبرمج وبيبقوا تابعين للغة بشكل أساسى دول أسمهم المكتبة الرئيسية للغة Standard Library.

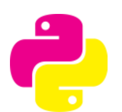
مثلاً لو عاوز تجيب المضروب بتاع رقم معين.. هل لازم تكتب الكود اللى بيجيب المضروب زى ما عملنا ولا اللغة بتوفرلك مكتبات فيها دوال تساعدك تحسبه بسهولة! 🤔

الحقيقة البايثون فيها مكتبة Math اللى أتكلمنا عنها قبل كده و فيها دوال بتعمل الكلام ده من غير ما تكتبه وقس على هذا الموضوع فى كل التطبيقات اللى بتعملها هل لازم تبدأ من الصفر from scratch ولا فيه مكتبات مساعدة.

يبقى الـStandard Library التابعة للغة بتوفرلك مكتبات/موديلوز فيها دوال وكلاسات تعملك أغلب الوظائف اللى بتحتاجها.

وممكن تشوف الوثيقه فى الموقع الرسمى للغة وتشوف محتوي الـStandard Library.

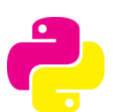
المكتبة الرئيسية فيها نوعين من الموديلوز..



- الأول built in modules ودى أتكتب بلغة الـ c أثناء تصميم اللغة نفسها وهى أصل اللغة واللى بتتعامل مع نظام التشغيل بشكل مباشر.
- والنوع التانى موديلوز إتكتبت بلغة البايثون بعد ما إتعملت اللغة عشان تسهل عليك إنك تعمل التاسكات البرمجية المختلفة ودى تقدر تفتحها وتشوف أكواد البايثون فيها.

من أهم الحاجات برده فى الـ standard Library للغة البايثون واللى أتكلمنا عنها قبل كده الدوال built in functions هتلاقى إننا استخدمنا عدد كبير منهم لحد دلوقتى وهتحتاج بعضهم بشكل متكرر فى كل السكريتات بتاعتك.

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	



بعض الموديولات المشهورة

فى الجزء ده هنتكلم عن 3 من الموديولات المشهورة فى لغة البايثون التابعين للـ standard library وهم stmplib ,sys,time

- قولنا إن المكتبات دى فيه منها حاجات تابعة للـ Interpreter نفسه يعنى متقدرش تشوف الكود بتاعها وفيه حاجات مكتوبين بلغة البايثون وتقدر تشوف السكريتات py. بتاعتها.بالنسبة للـ modules المكتوبين بلغة البايثون بيبقوا موجودين فى المجلد Lib التابع لمجلد اللغة وهنعرف أزاى نجيب مكان الموديول برمجياً.
- عشان تعرف الموديول ده مكتوب بلغة الـ c ولا عبارة عن سكريت مكتوب بلغة البايثون.. مثلاً هنجرب على موديول اسمه os مثلاً

```
>>>import os
>>> os.__file__
'C:\\Users\\Mahmoud\\AppData\\Local\\Programs\\Python\\Python36-32\\lib\\os.py'
```

- فى الكود ده عملنا import لموديول اسمه os وطلعنا منه متغير اسمه __file__ والمتغير ده بيحطه المفسر فى الموديولات المكتوبة بلغة الباثون لما تعملها import.
- وقيمة المتغير ده عبارة عن المسار اللى فيه الموديول وطبعاً المسار الغريب ده اللى متثبت عليه لغة البايثون الإصدارات < 3.6 على ويندوز 10
- لو إصدار قديم شوية مجلد python هتلاقيه فى المسار c:/python
- لو روحت للمسار بتاع الموديول os هتلاقيه سكريت بايثون os.py.
- ده يعنى ان الموديول os موجود ظاهرياً على شكل سكريت بايثون ومش built in.



- بالنسبة للموديولات الـ built in زى موديول math وده مش هتلاقيه فى مجلد lib .

```
>>> import math
>>> math.__file__
AttributeError: module 'math' has no attribute '__file__'
```

فى الجزء تعمدت أخليك تدور على مكان الموديول عشان تعرف أنه مهما راح او جه هتلاقيه عباره عن سكريبت بايثون فيه شوية دوال ومتغيرات مش حاجة تخوف يعنى 😁.

موديول time

تقدر من خلال لغة البايثون إن تعرض الوقت والتاريخ بطرق ودوال محدده من خلال الموديول time اللى بالمناسبة built-in زيه زى موديول math يعنى مش هتلاقيه فى الفولدر Lib.

أول حاجة عملنا import للموديول باسم t للتبسيط.	`>>> import time as t`
دالة time بترجع الوقت بالثوانى من تاريخ 1970/1/1 وده ده أحد تواريخ الوقت الموحد فى العالم UTC واللى بتعتبره بعض أنظمة التشغيل ريفرنس عشان تحسب الوقت من بعده.	`>>> t.time()` `1538089706.9603162`
دالة ctime بتاخد برامتر واحد وهو الوقت معبراً عنه بالثوانى وتحوله إلى صورة التاريخ والوقت المتعاده. لاحظ لما كان البرامتر 0 جابت تاريخ UTC بتاع 1970. لو عاوزها تجيب التاريخ و الوقت الحالى بتمرر لها الثوانى اللى بترجعها دالة time() أو من غير أى برامترز.	`>>> t.ctime(0)` `'Thu Jan  1 02:00:00 1970'` `>>> t.ctime(t.time())` `'Fri Sep 28 02:19:30 2018'` `>>> t.ctime()` `'Fri Sep 28 02:19:31 2018'`



`>>> x = t.localtime()` `>>> x` `time.struct_time(` `tm_year=2018, tm_mon=9,` `tm_mday=28, tm_hour=2, tm_min=27,` `tm_sec=49, tm_wday=4,` `tm_yday=271, tm_isdst=0)` `>>> print('Now %sh:%sm:%ss'%` `(x.tm_hour,x.tm_min,x.tm_sec))` `Now 2h:27m:49s`	دالة localtime بترجع time_struct اللى من خلاله تقدر تطلع أجزاء التاريخ زى السنه tm_year والشهر tm_mon واليوم tm_mday والوقت زى الساعه tm_hour والدقايق tm_min والثوانى tm_sec.

عشان تعرف باقى الدوال فى موديول time تقدر تشوف الوثيقه من هنا time.



موديول sys

من الموديولات المهمة واللى بتحتاج تستخدمها فى سكريبتاتك فكان لازم أتكلم عنها..

موديول sys فيه مجموعة من المتغيرات اللى بتحمل معلومات مهمة عن لغة البايثون ونظام التشغيل زى ما هنشوف.

`>>> import sys`	اسم الموديول sys
`>>> sys.builtin_module_names` `'_ast', '_bisect', '_blake2', '_codecs', '_codecs_cn',...)`	المتغير بيرجع tuple فيها كل اسامى الـ built in modules واللى منهم math و time والموديول نفسه sys
`>>> sys.modules` `>>> 'os' in sys.modules` `True`	المتغير modules بيرجع dict كبيره فيها كل الموديولز الـ built-in و الموديولز العاديه. تقدر عن طريق الأمر in تعرف إذا كان الموديول موجود أو لا.
`>>> sys.path`	path بيرجع list فيها كل الاماكن اللى ممكن يقرى منها الموديولز ويعملها import.
`>>> sys.platform` `'win32'`	platform بيرجع نوع نظام التشغيل لو ويندوز win32 لو لينكس linux وهكذا.
`>>> sys.version` `'3.6.3 (v3.6.3:2c5fed8..'`	version بيرجع إصدار البايثون الحالى.

و عشان تعرف أكتر عن المتغيرات والدوال فى موديول sys تقدر تكمل من هنا sys



أستنى لسه مخصلناش كلام عن موديول sys 😁 ومن الاستخدامات المهمة جداً لموديول sys إنه بيخليك تقدر تمرر قيم للسكريت بتاعك عن طريق الـ command line.

- أنت عارف إنك لو عاوز تدخل قيمة للسكريت بيكون عن طريق دالة input ولكن فيه طريقة تانية.
- هتعمل سكريت جديد هتسميه أى اسم..انا سميته sys_argv.py مثلاً..

```
from sys import argv
try:
    script=argv[0]
    user  =argv[1]
    print('script name :',script)
    print('user name :',user)
except:
    print('no inputs from command line!')
```

Download script sys_argv.py from GitHub

- لو عملت run للسكريت فى أى مكان هيحصل exception وهقولك ليه.. وهتظهر الرسالة no inputs from command line.
- والحل إنك تعمله run فى الـ command line وتمرر له برامترات بعد اسم السكريت بالشكل ده python script.py value1 value2 value n.. بالطريقة دى كل القيم اللى هتكتبها ومفصول بينها بمسافة وحتى اسم السكريت بعد كلمة python هتتحط فى list اسمها argv.



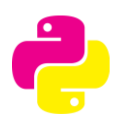
```
E:\>
E:\>python sys_argv.py
no inputs from command line!

E:\>python sys_argv.py Hello World
script name : sys_argv.py
user message : Hello

E:\>python sys_argv.py Hello_World
script name : sys_argv.py
user message : Hello_World
```

- لما كتبت اسم السكريبت بس .. كان argv[0] اسم السكريبت ومفيهاش argv[1] فحصل exception.
- لما كتبت بعد اسم السكريبت Hello World وكان فيه مسافة بين الكلمتين فبقى argv[0] اسم السكريبت و argv[1] كلمة Hello وargv[2] كلمة World عشان كده متطبعتش غير كلمة Hello.
- يعني لو عاوز أبعت أى قيم للسكريبت من خلال الـ command line بكتب القيم دى بعد اسم السكريبت مفصول بينها بمسافة وبعمل import للـ list argv من موديول sys وبطلع منها القيم.

الطريقة دى حلوة لو عاوز تبعت قيم للسكريبت بتاعتك من غير ما تستخدم دالة input وهتلاقيها مستخدمه فى برامج كتير قد تكون تعاملت مع بعضها لو متعاملتش فأديك عرفت أو فهمت البرامج دى بتاخد الـ inputs ازاى من الـ command line حتى لو كانت معمولة بلغات تانيه.



موديول smtplib

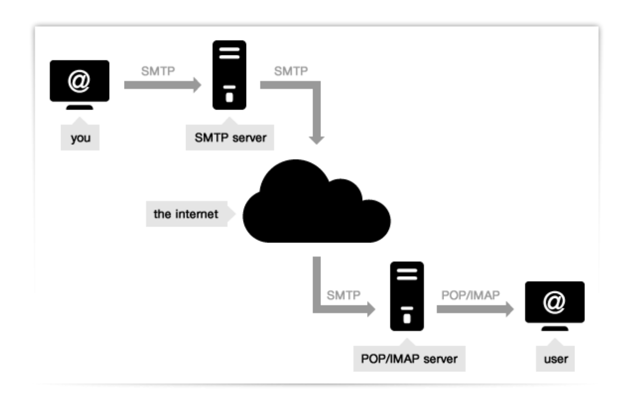
موديول smtplib أحد الموديولات اللى بنستخدمها فى إرسال إيميلات بلغة البايثون..

و stmp ده البروتوكول الخاص بإرسال البريد الإلكترونى وتقدر تعرف عنه أكتر من هنا

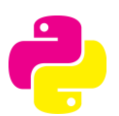
Simple_Mail_Transfer_Protocol

طريقة إرسال الإيميل

لو عندى مستخدم A عاوز يبعت إيميل لمستخدم B العملية هتتم أزاى🤔؟



- فى البداية A لازم يكون مستخدم معروف لدى الـ mail server زى ما تكون مسجل إيميل على ياهو أو جيميل..وبالتالى بالإيميل ده تقدر تبعت وتستقبل الرسايل.
- لو المستخدم A بيبعت رساله إلى B.. الرسالة بتروح للـ mail server الخاص بيه عن طريق بروتوكول smtp .

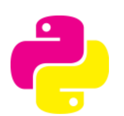


- الـ mail server هيشوف عنوان الإيميل اللى مبعوت ليه الرسالة موجود على نفس السيرفر فيبعتها له على طول ولا على سيرفر تانى فيبعتها للسيرفر ده.
- مثل أنا عندى إيميل ياهو لما أبعت الرسالة هتروح للسيرفر بتاع ياهو smtp.mail.yahoo.com وبعدين السيرفر ده هيشوف أنا باعت الرسالة دى لإيميل yahoo فيبعتها له على طول ولا رايحه لإيميل gmail مثلاً فيبعتها للسيرفر بتاع جيميل smtp.gmail.com .
- عملية إستقبال الرسايل بتتم ببروتوكول زى POP post office protocol..

أعرف اكتر عن الموضوع ده من هنا what-is-smtp-server

يعنى عشان تبعت إيميل بلغة البايثون هتحتاج إيميل yahoo أو gmail والميل سيرفر بتاعهم. ومن تجربتى إتضح ليا إن استخدام الـ yahoo account أسهل من gmail وهتعرف بعد شويه ليه وطبعاً انت حر تستخدم اللى يناسبك..

- المهم فى الأول هتسجل إيميل على yahoo أو gmail أو أى موقع بيقدم خدمة البريد الإلكترونى.
- مكتبة stmplib بتحتاج عنوان الإيميل اللى سجلت بيه والباسورد بتاعه بالإضافة لعنوان الـ mail server للموقع اللى سجلت عليه.
- وطبعاً لما الموقع بتاع الإيميل يلاقى برنامج غريب عمل تسجيل دخول هيفتكره محاولة أختراق للحساب و وأقل واجب يعمله بلوك ويمنعه من إرسال إيميلات وده اللى بيحصل



- فبحتاج تفعل خاصية فى الإيميل بتسمح للأجهزة والتطبيقات إنها تسجل دخول على الإميل بتاعك Let less secure apps access your account.

بالنسبة لإيميل ياهو من خلال الـ security او أمان الحساب security

Allow apps that use less secure sign in

Some non-Yahoo apps and devices use less secure sign-in technology, which could le account vulnerable. You can turn off access (which we recommend) or choose to use t despite the risks.

Learn more

السماح للتطبيقات التي تستخدم تسجيل دخول أقل أمانًا

تستخدم بعض التطبيقات والأجهزة غير التابعة لـ Yahoo تقنية تسجيل دخول أقل أمانًا، وهذا قد يعرض حسابك للاختراق. يمكنك إيقاف إمكانية الوصول (وهذا ما نوصي به) أو اختيار استخدامها بالرغم من المخاطر.

اعرف المزيد

- بالنسبة لإيملات الـ Gmail طريقة تفعيل الخاصية من هنا وممكن تفعل الخاصية دى وبرده يعمل بلوك للطلبات اللى بيعملها السكريت بتاعك وميخلهوش يبعت إيـميلات..عشان كده بفضل موقع ياهو.

- بعد كده تعرف عنوان الـ smtp server بالنسبة لياهو smtp.mail.yahoo.com وبالنسبة لـجيميل smtp.gmail.com وللمزيد من هنا https://serversmtp.com

دلوقتى عملنا إيميل على أى موقع وعرفنا الـ mail sever بتاعهم وجهزنا الإيميل إنه يبقى متاح لأى برنامج معاه الباسورد واليوزر بتوعه أنهم يبعتوا ايميلات من خلاله.. والخطوة الجاية هنبعت الإيميل بمكتبة stmplib.

- أول حاجة هتعمل import للكلاس stmplib اللى هتبعت الإيميل والكلاس EmailMessage اللى بتاخد منها object بتحط فيها اجزاء الإيميل زى from,subject,body اللى بتبعتهم مكتبة stmplib.



```
import smtplib
from email.message import EmailMessage
```

بعد كده هتعمل أعدادت الـ mail server والـ account بتاعك.

```
server =smtplib.SMTP(server,port)
server.starttls()
server.login(your_login_email,your_password)
```

- دالة STMP بتاخد أول برامترين عنوان السيرفر..لو كنت سجلت في yahoo وبيكون فهو smtp.mail.yahoo.com لو Gmail فهو stmp.gmail.com.
- والبرامتر التانى هو رقم البورت واللى بيكون 587 فى حالة استخدامك لتشفير TLS transparent security layer وده من وسائل التشفير للإيميل فى عملية الإرسال.
- وطبعاً لو حددت البورت 587 لازم تفعل TLS بالدالة starttls.
- دالة login بتاخد عنوان الإيميل بتاعك اللى سجلته وفعلت فيه خاصية less secure application access والباسورد بتاعه.

بعد كده هنجهز الرسالة اللى هنبعتها.

```
msg = EmailMessage()
msg['From'] = your_login_email
msg['To'] = the_reciever
msg['Subject'] = "Python email"
body = "Python test mail"
msg.set_content(email_body)
```



- بتعمل object من الكلاس EmailMessage اللى بتحط فيه اجزاء الرسالة.. وخلى بالك إن from لازم تكون الإيميل اللى عملت بيه تسجيل دخول فى داله login.

أخيراً بتبعت للرسالة بدالة send_message وبتقفل السيرفر بدالة quit.

```
server.send_message(msg)
server.quit()
```

من الخطوات اللى فاتت عملت موديول mail.py وده بيجمع كل الخطوات اللى فاتت.

```
import smtplib
from email.message import EmailMessage
def send(con,fields):
    server =smtplib.SMTP(con['server'],con['port'])
    server.starttls()
    server.login(con['from'],con['pass'])

    msg = EmailMessage()
    msg['From']      = con['from']
    msg['To']        = con['to']
    msg['Subject']   = con['subject']
    body=''
    for key,val in fields.items():body+='%s : %s\n'%(key,val)
    msg.set_content(body)
    result=server.send_message(msg)
    server.quit()

    if result=={}:return True
    else:return False
```

Download script mail.py from GitHub



كل الفكرة إني عملت دالة وسميتها send بتاخد برامترين.. الاول con وده dict فيه بيانات الإتصال mail server,email,password والحاجات دى والبرامتر التانى dict برده فيه الرسالة أجزاء الرسالة وتعالى نشوف مثال لاستخدام الموديول ده.

```
from mail import send
con={'server'  : 'smtp.mail.yahoo.com',
     'port'    : '587',
     'from'    : '...@yahoo.com',
     'pass'     : '*****',
     'to'      : '...@gmail.com',
     'subject' : 'User'
    }
fields={'name':'Python Programmer',
'Age':23,
'phone':'1234',
'message':'I Love Python ❤'
}

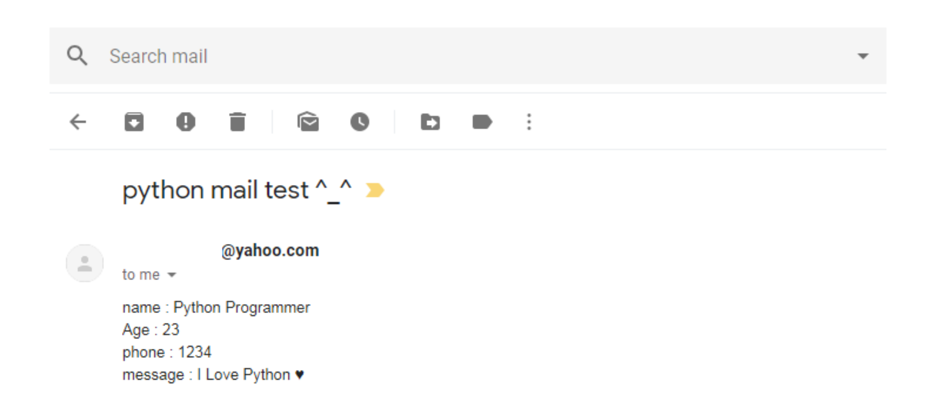
if send(con, fields):print('Done..')
else : print('failed..')
```

Download script send_email.py from GitHub

- هنا عملت import للدالة send فى للموديول mail.py.
- فى الـ dictionary اللى اسمها con أنا استخدمت mail server وأكونت بتوع ياهو وحدد الإيميل اللى هتتبعتله الرسالة to طبعاً متنساش تستخدم الأكونت والباسورد الخاصيين بك.
- فى الـ dictionary التانيه fields دى أجزاء الرسالة.



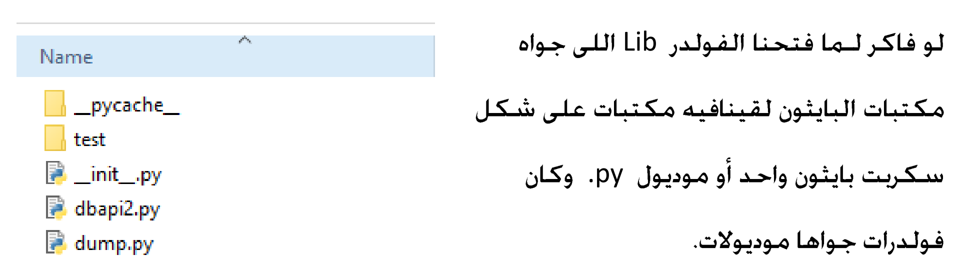
ده شكل الرسالة لما وصلت على أكونت Gmail



وأخيراً لو عاوز امثلة اكتر وازاى تبعت مرفقات attachment مع الإيميل أو صفحات بتنسيق html زى الإعلانات اللى بتجيلك على الإيميل تقدر تكمل من هنا email.examples



الحزم Packages.



لو فاكر لما فتحنا الفولدر Lib اللى جواه مكتبات البايثون لقينافيه مكتبات على شكل سكريت بايثون واحد أو موديول py. وكان فولدرات جواها موديولات.

الفولدر ده اسمه حزمه Package.. وهو عبارة عن اكتر من موديول يتم جمعهم فولدر.

ليه بنعمل الـPackage ؟!

- لما تبقى المكتبة كبيرة ومن الأفضل إنك تقسمها على أكثر من موديول كنوع من التنظيم.
- لما تعمل مشروعك أو مكتبتك فى Package بتبقى مضغوطة بصيغة معينة زى whl. وتقدر تثبتها بسهولة أو تنشرها وأى حد يستخدمها.

لو بتعمل أى مشروع وهتحتاج لمكتبات تساعدك .. هتبحث وتشوف المكتبات الموجودة فى الـ standred library هل كفاية ليك ولا محتاج تحمل مكتبات من على الأنترنت.. وساعتها هتدور على packages تثبتها على جهازك.



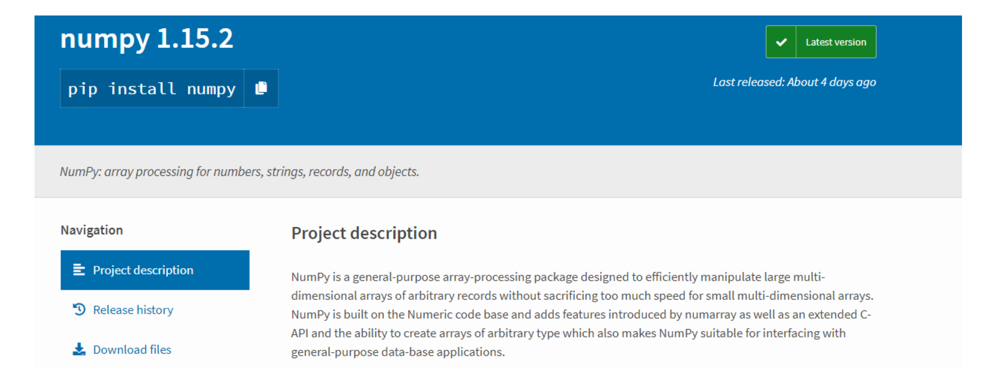
موقع PyPi

الموقع ده المستودع الأكبر لكل المكتبات والـ Packages للغة البايثون اللى من خلاله تقدر تبحث وتحمل المكتبات اللى أنت عايزها ولو ربنا فتح عليك وعملت مكتبة تقدر تنشرها من خلاله والمستخدمين يحملوها pypi.org

Find, install and publish Python packages with the Python Package Index

Search projects

تعالى ندور على اسم أى مكتبة ونشوف هنثبتها عندنا أزاى ...مثلاً مكتبة numpy ودى مكتبة بتخليك تستخدم المصفوفات متعددة الأبعاد multi-dimensional arrays ودى من الحاجات المش متوفرة فى الـ standard library فى لغة البايثون وهنحتاجها وهنتكلم عنها فى الفصل اللى جاى فى موضوع معالجة الصور... المهم هتبحث عن اسمه numpy.





طبعاً طبيعتك الفطرية هتروح ناحية download files وهتحمل ملفات المكتبة وأبقى وريني هتثبتهم ازاى 😂.

```
pip install numpy
```

الحقيقة بايثون عملت طريقة اسهل من كده اسمها pip ولعلك تكون لمحت الامر اللى من فوق .. تعالو هنا عشان نعرف ايه الـ pip ده.

أداة PIP

الـ pip دة اداة او نظام لإدراة المكتبات والـ packages فى لغة البايثون من خلاله تقدر تثبت وتحدث المكتبات وتلغى تثبيتها وتعتبر طريقة سهلة وعملية جداً وهتوفر عليك كتير.

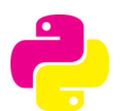
pip بتتعامل معاه من الـ command line يعنى لازم تكون بتعرف تتعامل مع CMD أو Terminal كما يسمى فى نظام لينكس.

من الجميل إنك مش بتحتاج تثبت برنامج الـ pip لانه بيجى مع لغة البايثون فتقدر تتأكد إنه موجود بأنك تفتح الـ command line وتكتب فيه pip –version ..

```
C:\Users\Mahmoud>pip --version
pip 10.0.1 from c:\users\mahmoud\appdata\local\programs\python\python36-32\lib\site-packages\pip (python 3.6)
```

لو مش موجود الـ command line هيقولك انت بتقول أيه 😂 وتعرف إنك متحاج تثبته بنفسك وياريت متكسلش عشان هيفيدك جداً.

ودى طريقة تثبيته installing pip.



أوامر PIP

- عشان تثبت مكتبة او Package بتكتب pip install + اسم المكتبة اللى هتعرفه من موقع PyPi او هتنسخ الأمر كله من هناك

```
pip install package_name
```

- عشان تعمل تحديث للـ Packages بتكتب الامر --upgrade قبل اسم المكتبة..

```
pip install --upgrade package_name
```

- عشان تمسح الـ Package.

```
pip uninstall package_name
```

- عشان تحدث الـ pip نفسه لأنك لو شغال على إصدار قديم هتحصل معاك بعض المشاكل

```
pip install --upgrade pip
```

طبعاً عارفنى مش هجيب كل حاجة من الـ documentation يا دوب الكلمتين اللى هينفعونا وعشان يقولوا أننا جامدين وبنستخدم الـpip 😁

لو انت عاوز تكمل وتشوف باقى الأوامر من هنا quickstart .. لكن أعلم إنى هجبلك الخلاصة.



استخدام PIP

قولنا عاوزين نثبت مكتبة numpy .. هنروح موقع pypi

```
pip install numpy
```

تانى ناخد منه اسم المكتبة أو تنسخ الأمر كله وتحطه فى الـ command line عشان يبدأ التثبيت.

```
C:\Users\Mahmoud>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/ff/e9/7ee1eefad3ac289cf609c2b9305afe6362f75855
/numpy-1.15.2-cp36-none-win32.whl (9.9MB)
    100% |████████████████████████████████| 9.9MB 20kB/s
Installing collected packages: numpy
Successfully installed numpy-1.15.2
```

بعد ما تتثبت المكتبة ويقولك successfully installed بالمرة عشان تتأكد أنها أتثبتت وتمام..أفتح الـ python interpreter فى الـ command line وأعملها import.

```
C:\Users\Mahmoud>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.__file__
'C:\\Users\\Mahmoud\\AppData\\Local\\Programs\\Python\\Python36-32\\lib\\site-packages\\numpy\\__init__.py'
```

وأستخدمنا المتغير __file__ عشان أعرفك مكان الـ Package أتثبتت فين..المكتبات اللى بتثبتهم عن طريق pip بيروحوا فى مجلد اسمه site-packages جوه مجلد lib.

عمل مكتبة Packaging

هتسألنى فى الأول أنا هعملها ليه أصلاً 🤔؟ .. هتعملها عشان لما ربنا يكرمك تقدر تعمل مكتبة مهمة وترفعها على موقع pypi وده الموقع الرسمى اللى هتحمل منه المكتبات والحزم اللى هتحتاجها فى مشاريعك.



قولنا المكتبة عبارة عن مجموعة من الموديلوز modules فى مجلد بيتم ضغطه بطريقة معينة عشان تترفع على موقع pypi أو أى موقع والمستخدم العادى يحملها يعملها install عن طريق pip.

عاوزين نعمل Package قابلة للتثبيت للموديول factorial.py اللى علمناه عشان يحسب مضروب العدد.

```
-factorial
   -__init__.py
-setup.py
-LICENSE
-README.md
```

- هتعمل فولدر وتسميه facotrial وهتحط جواه السكريبت factorial.py اللى هتغير اسمه وتخليه تسميه __init__.py
- لما تحط سكريبت __init__.py جوه فولدر ده بيعرف الـ interpreter إن الفولدر ده فيه موديولات للغة البايثون.
- ولما تعمل import للموديول بيكون كده import factorial وفى الحالة دى هيعمل import للسكريبت __init__.py اللى جوه الفولدر factorial..
- بره الفولدر facotrial هتعمل ملف README.md (مش ضروري) ولو كنت من مستخدمين GitHub هتعرف إن الملف ده بيتكتب بتنسيق معين اسمه Markdown وده بيتحط فيه وصف وشرح المكتبة.
- هتعمل ملف LICENCE (مش ضرورى) وده بيتحط فيه رخصة المكتبة.
- ملف setup.py وده مهم جداً وده اللى فيه أعدادات تثبيت المكتبه.

لما تعمل تثبيت للمكتبة بتاعتك عن طريق pip .. هيستنسخ المجلد factorial لفولدر lib/site-packages ولما تعمل استدعاء للمكتبة هيكون بالشكل ده import factorial.



- هتيجى فى المجلد الرئيسى وهتعمل ملف README.md اللى هيكون بالشكل ده

```
#Factorial
factorial has magic function to calculate the factorial of any numbers ^_^
```

- لما ترفع المكتبة بتاعتك على GitHub مثلاً ملف readme ده زى الـ documentation اللى بتعرف الناس المكتبة دى بتعمل ايه .. فى تنسيق الـ md أي سطر يبدأ بـ # مش تعليق زى البايثون انما بيبقى Header أو عنوان حجمه اكبر زى tag <h1> فى لغة html وبرده لو بتشتغل على GitHub هتكون فاهم الحوار 😁.
- هنرجع للملف setup.py وهو ده الأهم هتحط فيه الكود ده ..

```
import setuptools
long_description = open("README.md", "r").read()
setuptools.setup(
    name="factorial",
    version="0.0.1",
    author="Mr python",
    author_email="author@example.com",
    description="A small example package",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="anyurl",
    packages=setuptools.find_packages(),
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
)
```

Download script setup.py from GitHub

- اللى بيعمل import لمكتبة setuptools ودى مكتبة فيها مجموعة موديولات بتسهل عليك تخزيم وعمل الـ Packages.



- فى البداية بيقرى النص اللى فى ملف README وبيحطه فى متغير long_description عشان متسألش السطر ده بيعمل ايه.

- باقى المتغيرات .. name ده اسم الـ package بتاعتنا احنا خليناه factorial و version ده رقم النسخة .. Author اسمك كصاحب الباكدج ..

- بعدين متغير الوصف description والوصف الطويل long_description اللى هو ملف README ..

- متغير packages ده بتحدد فيه المجلد أو المجلدات اللى فيها ملفات المكتبة بتاعتك وعندك طريقتين . إما تحدد اسم المجلد اللى فيها ملفات البايثون اللى هو package بالطريقة دى ['packages = ['factorial .. أو تخلى مكتبة setuptools تدور بنفسها على أى فولدر فيه ملف اسمه __init__.py وتعتبره هو المكتبه بدالة find_package.

كده جهزنا المكتبة بتاعتنا ناقص نعملها packaging.. الخطوة الجاية لازم تكون بتعرف تشتغل على الـ command line عشان تقدر تعمل تحزيم للمكتبة ..

- فى الأول عن طريق pip هتعمل تحديث للمكتبات setuptools و wheel.

```
pip install --upgrade setuptools wheel
```

- طبعاً شوفت السكريت setup.py وعرفت إن setuptools دى بتحدد فيها ملفات وبيانات المكتبة اللى عاوز تعملها Packaging..أما wheel ودى بتضغط للمكتبة أو



بصيغة whl. عشان تبقى ملف واحد جاهز للرفع على موقع PyPi ونشرها لأى حد يحملها ويثبتها.

- بعدين أفتح الـ command line على المسار اللى فيه ملفات المكتبة ونفذ الأمر ده.

```
python setup.py sdist bdist_wheel
```

..أنا مثلاً حطيت الملفات فى فولدر اسمه package فى البارتيشن E فدخلت للمسار ده ونفذت الأمر.

```
E:\>cd package

E:\package>dir
 Volume in drive E is Mahmoud
 Volume Serial Number is 6BFA-A0A0

 Directory of E:\package

09/28/2018  11:15 PM    <DIR>          .
09/28/2018  11:15 PM    <DIR>          ..
09/28/2018  11:11 PM    <DIR>          factorial
09/28/2018  11:13 PM                87 README.md
09/28/2018  11:13 PM               573 setup.py
               2 File(s)            660 bytes
               3 Dir(s)  21,981,188,096 bytes free

E:\package>python setup.py sdist bdist_wheel
```

- لو ظهرت عندك أخطاء تبقى معلتش الخطوة اللى فاتت بتاعة تحديث wheels ولو العملية تمت بنجاح .. هتفتح الفولدر هتلاقى اتولد فيه فولدرين build و dist .
- جوة الفولدر dist هتلاقى الملف whl. وده بيمثل المكتبة بتاعتك جاهزة للرفع على موقع PyPi وأى حد يقدر يحملها ويثبتها..بس الاول تعالى نثبتها احنا عندنا ونجربها 😁

- هتدخل الـ command lineجوه الفولدر dist هتلاقى فيه الـ Package وهو ملف whl. اسمه زى كده factorial-0.0.1-py3-none-any.whl.



- هتعملـه install بالـ pip install + اسم الملف بتاع الباكـدج.

```
E:\package>cd dist

E:\package\dist>dir
 Volume in drive E is Mahmoud
 Volume Serial Number is 6BFA-A0A0

 Directory of E:\package\dist

09/28/2018  11:17 PM    <DIR>          .
09/28/2018  11:17 PM    <DIR>          ..
09/28/2018  11:17 PM             1,475 factorial-0.0.1-py3-none-any.whl
09/28/2018  11:17 PM             1,084 factorial-0.0.1.tar.gz
               2 File(s)          2,559 bytes
               2 Dir(s)  21,981,188,096 bytes free

E:\package\dist>pip install factorial-0.0.1-py3-none-any.whl
Processing e:\package\dist\factorial-0.0.1-py3-none-any.whl
Installing collected packages: factorial
Successfully installed factorial-0.0.1
```

لـو التثبيت تم بنجاح هيقولك successfully installed.وعلـى طول تفتح

الـ python interpreter وجّرب المكتبة العظيمة بتاعتنا 😂.

```
E:\package\dist>python
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import factorial
>>> factorial.fact(6)
720
>>> print(factorial.__file__)
C:\Users\Mahmoud\AppData\Local\Programs\Python\Python36-32\lib\site-packages\factorial\__init__.py
>>>
```

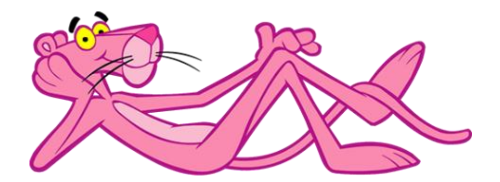
وطبعاً بالمتغير__file__ جبتلك المسار اللى راحت فيه ملفات المكتبة بعد التثبيت وهو lib/site-packages/factorial.

تقدر من هنا تحمل ملفات الـ Package

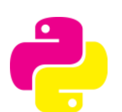
وللمزيد عن الموضوع ده وعشان تعرف أزى ترفع الـ Packages بتاعتك على موقع PyPi كمل من الروابط ده packaging-projects و Distutils - Tutorial



الفصل 4 – تخزين البيانات Data Storages



الفهرس



التخزين فى المتغيرات Variables

لحد الآن التخزين بتاعنا على مستوى الرامة.. نقدر نخزن فيها متغيرات بأنواعها ونسترجعها فى أى وقت بس لما نقفل البرنامج كل حاجة بتروح.. لكننا عاوزين نخزن الداتا بشكل دائم وبطريقة منظمة بحيث إننا نقدر نسترجعها فى اى وقت وتفضل موجودة بعد ما نقفل البرنامج.

عشان كده عملوا طرق تخزين البيانات المختلفة اللى منها ممكن تخزن الداتا فى ملفات Files او فى قواعد بيانات database تكون موجودة على الجهاز بتاعك أو موجودة على سيرفر تانى وتتصل بيها اونلاين.

واستخدامك للطرق دى يعتمد على شكل الداتا اللى عاوز تخزنها وحجمها وطبيعة البرنامج بتاعك.

- فى الجزء ده هديك مشكلة وهندور ليها على حلول مع بعض وهنشوف أيه أنسب حل ليها واللي ينفع تستخدمه فى تطبيقاتك المختلفة.

student	class	section	percentage
ahmed	1	a	90
aly	2	b	80
adam	3	c	70

- اللى قدامك ده جدول بسيط جداً فيه اسم كل طالب والفرقة بتاعته والسكشن والنسبة المئوية.



- عاوزين نعمل سكريت بلغة البايثون يطلب مننا إدخال الاسم والصف والسكشن والنسبة لكل طالب وبعدين نقدر نبحث عن كل طالب باسمه ونشوف البيانات بتاعته اللى خزناها.

سيبك من التخزين دلوقتى وتعالا نشوف السكريت شكله ايه..

فى الأول إيه الـ data type المناسب اللى أخزن فيه البيانات بتاعة الـ students؟ 🤔 الأنسب هو dictionary.. فعملنا dict اسمه students

```
students={}
```

- بعدين هنعمل دالة ونسميها insert ودى هتستقبل من اليوزر عن طريق دالة input بيانات كل طالب

```
def insert():
  while True:
    name=input("Student Name : ")
    _class=input("Student Class : ")
    section=input("Student Section : ")
    percent=input("Student Percentage : ")
    student={'class':_class,'section':section,'percent':percent}
    students[name]=student
    agin=input("Any More ? ")
    if(agin!='yes'):break
```



- لما تعمل call للدالة insert هتدخل فى جملة while loop عشان تقدر تدخل بيانات أكتر من طالب.
- عن طريق دالة input هنقرى من المستخدم name وclass_ وتلاحظ عملت _ قبل المتغير لأن كلمة class من الكلمات المحجوزة.. و section وpercent..وهيتحطوا فى dict برده كل key والقيمة بتاعته.
- الـ dict دى هتتخزن فى students بالـ key اللى هو اسم الطالب name.
- أخيراً دالة input هتسألك any more.. لو عاوز تدخل بيانات طالب تانى هتكتب yes هتروح فى المتغير again اللى لو مكنش yes هيعمل break ويطلع من التكرار وبكده نكون خلصنا نعمل insert.

ده شكل الدالة لما تعملها call وتدخل البيانات.

```
Student Name : Ahmed
Student Class : 1
Student Section : A
Student Percentage : 90
Any More ? yes
Student Name : Ali
Student Class : 2
Student Section : B
Student Percentage : 80
```

وده شكل students بعد ما دخلنا فيها بيانات اتنين طلاب

```
{'Ahmed': {'class': '1', 'section': 'A', 'percent': '90'},
'Ali': {'class': '2', 'section': 'B', 'percent': '80'}}
```



- بعدين عملنا دالتين search دى عشان نبحث بيها عن بيانات الطالب و remove عشان نمسح الطالب من students.

```
def search():
    name=input("Enter Student Name : ")
    if name in students:
      print('Name :',name)
      for key,value in students[name].items():
        print(key,':',value)
    else : print('student not found')

def remove():
    global students
    name=input("Enter Student Name : ")
    if name in students:students.pop(name)
    else : print('student not found')
```

- الدالتين زى بعض..كل دالة بتقرى اسم الطالب عن طريق دالة input وتخزنه فى متغير name.. لو الاسم موجود name in students تعمل print لـ valueبتاعته اللى هـ بيانات الطالب فى حالة دالة search أو تمسحه بدالة pop فى حالة remove.

وده شكل البرنامج لما تعمل call للدالة search بعد ما سجلنا بيانات طالب اسمه Ahmed مثلاً.

```
Enter Student Name : ahmed
student not found
Enter Student Name : Ahmed
Name : Ahmed
class : 1
section : A
percent : 90
```



- أخر حاجة ربطنا الدوال فى جملة تكرار تطلب مننا أدخال 1 عشان نعمل insert أو2 عشان نعمل search و3 عشان نعمل remove و4 عشان يعرض كل الـ keys فى students واللى هم اسماء الطلاب إن وجدوا و0 للخروج من التكرار ده.

```
while True:
    x = input('1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : ')
    if  (x=='1') : insert()
    elif(x=='2') : search()
    elif(x=='3') : remove()
    elif(x=='4') : print(list(students.keys()))
    elif(x=='0') : break
```

السكريت كامل لو عاوز تحمله storages1.py وده شكل البرنامج لما تعمله run

```
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 1
Student Name : Mahmoud
Student Class : 3
Student Section : C
Student Percentage : 97
Any More ? yes
Student Name : Noor
Student Class : 1
Student Section : B
Student Percentage : 88
Any More ? no
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 4
['Mahmoud', 'Noor']
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 2
Enter Student Name : Mahmoud
Name : Mahmoud
class : 3
section : C
percent : 97
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 3
Enter Student Name : Noor
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 4
['Mahmoud']
```



لحد الآن وكل حاجة تمام وبنخزن ونمسح ونبحث.. بس المشكلة إن البرنامج بتاعنا لما نقفله ونعمله run من جديد مش بنلاقى الـ students اللى خزناهم عشان بيحفظهم فى الرامه فترة تشغيل البرنامج ولما تقفله على طول المتغيرات بتروح.

الحل إننا نستخدم إحدى طرق التخزين الدائم اللى هتكون اما الملفات Files او قواعد بيانات Databases .

الملفات Files

فى الجزء ده هنشرح إزاى تخزن الداتا فى ملف على الجهاز وزاى تسترجعها تانى وزاى تنشئ وتمسح وتعمل اعادة تسمية للملفات والمجلدات.

File Object

عشان تحفظ أو تقرى الداتا بتاعتك من ملف لازم تعمل file object وبأختصار ده اوبجكت لكلاس built in اسمها TextIOWrapper بتحدده من خلاله اسم الملف اللى هتتعامل معاه والـ access mode او طريقة تعاملك مع الملف ده وطريقة تخزين الداتا فيه بالإضافة إن الـ object ده فيه الـ methods اللى من خلالهم هتقرى وتخزن الداتا read,write.

عشان تعمل file object في لغة البايثون عندك اكتر من طريقة.. منها دالة open في موديول io أو الاختصارة ليها الدالة open مباشرة ودى built-in function.



```
import io
file_object = io.open(file_name, access_mode)
```

```
file_object = open(file_name, access_mode)
```

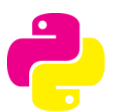
- البرامتر الاول file_name اسم الملف اللى هتخزن فيه الداتا أو هتقرى منه
- البرامتر التانى access_mode طريقة وصولك للملف وتشفير الداتا فيه.

- بالنسبة للـ access mode بتحدد من خلاله إنت عاوز تفتح الملف ده عشان تقرى منه read ولا تخزن فيه write.
- ولو مثلاً هتخزن نص فالملف هنا اسمه text file والـ string اللى هيتخزن فيه هيكون معموله encoding بتكويد UTF-8 أو cp1252 على windows.
- الموضوع بيختلف لو عاوز تخزن صورة مثلاً فى الحالة دى الملف اسمه binary file وتخزين وقراءة الداتا من الملف ده بيكون فى binary modes زى ما هنعرف.

يبقى أنت بتحدد الملف ده text file ولا binary file من خلال الـ access mode.

ودى قيم الـ access mode اللى تقدر تستخدمها..

Access Mode	Usage
r	للقراءة فقط read only
w	للكتابة فقط write only
w+	للقراءة والكتابة read and write لكن فى الحالة دى لو الملف كان فيه نص وكتبت عليه نص جديد اللى فيه بيتمسح وبيعمل overwrite



a+	للقراءة والكتابة read and write وفي الحالة دى لو الملف فيه نص وعملت كتابه بيضيف تحته النص الجديد يعنى بيعمل append مش overwrite
rb	للقراءة فقط فى نظام binary
wb	للكتابة فقط فى نظام binary
wb+	للقراءة والكتابة فى نظام binary
ab+	للقراءة والكتابة (appending) binary

هنشتغل على الـ interactive interpreter

```
>>> x = open('E://test.txt','w+')
>>> x
<_io.TextIOWrapper name='E://test.py' mode='w+' encoding='cp1252'>
```

- عملنا file object اسمه x.. وعملنا ملف test.txt فى البارتيشن :E على ويندوز وده مسار معروف بالنسبه لى وطبعاً تقدر تعمل الملف فى أى مسار تقدر توصله.
- الـ access mode للقراءة والكتابة read and overwrite.
- لو روحت المسار اللى حددته..بمجرد ما عملت open هتلاقى ملف فاضى اسمه test.txt.
- لو كنت لسه فاتح الـ interpreter وحاولت تمسح الملف أو تعدل فيه مش هيسمحلك بكده لأنه مفتوح عن طريق مفسر البايثون وده هيعرفنا بأخر ميثود المفروض كنت عرفتك عليها وتابعة للـ file object وهى close

```
File_oject.close()
```



- أكتب فى المفسر x.close() وهيقفل الـ I/O stream المفتوح ولو حاولت تمسح الملف أو تعدل عليه هتقدر ومفيش مشاكل.

لو عاوز تعرف باقى الـ methods الموجود فى كلاس TextIOWrapper عن طريق دالة dir(x) .

```
>>> dir(x)
['_CHUNK_SIZE', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '_
'__eq__', '__exit__', '__format__', '__ge__', '__getattribute__', '__getstate__
, '__init__', '__init_subclass__', '__iter__', '__le__', '__lt__', '__ne__', '_
_reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__'
_checkClosed', '_checkReadable', '_checkSeekable', '_checkWritable', '_finalizi
'closed', 'detach', 'encoding', 'errors', 'fileno', 'flush', 'isatty', 'line_bu
', 'newlines', 'read', 'readable', 'readline', 'readlines', 'seek', 'seekable',
```

عشان متحفش منهم 😂 أنا مش عاوزك تخاف أحنا مش هنستخدم غير 5 دوال وهم write و read و tell و seek وclose وهقولك بيعملوا ايه ودول تقدر توصلهم من خلال الاوبجكت x او الـ file object اللى عملناه عشان لما نيجى نكتب حاجة جوة ملف نصى ونستخدم دالة write ونعملها بالشكل ده x.write() متقوليش ليه الدالة دى تبع المتغير x مش built-in زى دالة open مش تابعة لأى حد.

دالة write

```
>>> x = open('E://test.txt','w+')
>>> x.write('Hello Python ^_^')
16
>>> x.close()
```

- فى الأول فتحنا ملف اسمه test.txt فى البارتيشن E طبعاً تقدر تخلى المسار زى ما انت عايز و access mode بتاع الملف كان w+ عشان نقدر نقرى ونكتب فيه.
- و كتبنا فيه جمل Hello Python ^_^ بدالة write.



- روح للمسار وأفتح الملف هتلاقيه برده لسه فاضي 😁 الحقيقة لازم تعمل close للملف عشان المحتوي يظهر فيه.

test.txt - Notepad
File Edit Format View Help
Hello Python ^_^

بعد ما تعمل close أفتح الملف وهتلاقيه كده.

- لو كررت الخطوات مرة تانية وعملت

x.write(' Hello Again ^_^') وفتحت الملف

test.txt - Notepad
File Edit Format View Help
Hello Again ^_^

هتلاقي النص الجديد بس اللي موجود عشان عمل overwrite وده لأن الـ acces mode عاملينه w+ و لو عاوز يضيف النص الجديد على القديم غير الـ access mode وخليه a+.

دالة read

تمام دلوقتي احنا كتبنا في ملف نصي وقفلنا البرنامج وجينا عاوزين نسترجع الكلام اللي كتبناه تاني ده بنعملها بدالة read .

- هنفتح الملف test.txt المرة دي بالمود w+ للقراءة والكتابه أو للقراءة فقط r.. بس الأول اتأكد انه مش فاضي فيه الجمله اللي كتبناها.

```
>>> x = open('E://test.txt','r')
>>> x.read()
'Hello Again ^_^'
```



- لو حاولت تعمل read مرة تانيه لنفس الملف هيطلعلك نص فاضي مش هيرجع المحتوى بتاع الملف طيب ليه🤔؟

```
>>> x.read()
''
```

- لما عملت read لأول مره الـ interpreter خلى الـ cursor بتاعه أو الـ postion اللى بيقرى منه عند اخر الملف.. فلما عملت read تانى بدأ يقرى من الآخر فملقاش حاجة والحل طبعاً فى الدالتين الجايين😂.

دالة tell و seek

باختصار دالة tell بتقولك المفسر واقف فين أو هيبدأ يقرى الملف من أى postion.

لما تعمل open للملف وتعمل read لأول مره دالة tell هترجعلك 0 وبعد ما تعمل read هيوقف عند اخر حرف فى الملف فدالة tell هترجعلك طول الملف او عدد الحروف..

دالة seek دى بتاخد برامتر واحد وهو position وبتحدد المكان اللى عاوز توقف عنده وتعالى نشوف بالأمثله..

فى الأول عملنا read للملف ولما عملنا read تانى رجع نص فاضى.	`>>> x = open('E://test.txt','r')` `>>> x.read()` `'Hello Again ^_^'` `>>> x.read()` `''`
قولنا نشوف الـ position اللى بيقرى منه بدالة tell طلع واقف فى أخر الملف.	`>>> x.tell()` `15`



نقلنا الـ position لأول حرف بدالة seek وعملنا read فقرأ محتوي الملف كله.	>>> x.seek(0) 0 >>> x.read() 'Hello Again ^_^'
ونقدر بدالة seek نقرى جزء بس من الملف مش الملف كله باننا نحدد position بدالة seek	>>> x.seek(5) 5 >>> x.tell() 5 >>> x.read() ' Again ^_^'

كده تـمام ..ويارِيت لما تخلص لعب متنساش تقفل الملف اللّه يكرمك 😂

دالة readlines

الـ object file فيه دوال كتير أهمهم read و write و seek وtell ولو كنت مركز فوق هتبقى عارف فايدة كل وحده فيهم .

فيه دالة جملية اسمها readlines بس قبل ما أقولك دى بتعمل أيه.. روح أفتح الملف اللى كنا شغالين عليه وأفتح بالنوت باد وأعمل فيه أكتر من سطر أى كلام المهم عاوزينه يبقى multiline أتصرف😂

```
test.txt - Notepad
File  Edit  Format  View  Help
Hello Again ^_^
A
B
C
```

- دلوقتى قولنا دالة read بتقرى الملف كله أو بتحددلها المكان بدالة seek وهو تقرى من بدايته.
- لو عاوز تقراه كسطور lines كل سطر يكون عنصر فى list فده بتعمله بدالة readlines



```
>>> x = open('E://test.txt','r')
>>> lines = x.readlines()
>>> lines
['Hello Again ^_^\n', 'A\n', 'B\n', 'C\n', 'D']
>>> lines[0]
'Hello Again ^_^\n'
>>> lines[1]
'A\n'
```

موديول os

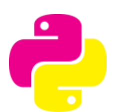
موديول os ده تبع الـ standard library و يعتبر موديول مهم جداً وعلى ذكر الملفات

موديول os فيه دوال تقدر تستخدمها فى مسح ونسخ وعمل وإعادة تسمية الملفات

والمجلدات الاول.

استدعاء الموديول	`import os`
دالة mkdir بتعمل مجلد جديد. هنا عملنا فولد فى البارتيشن E: اسمه folder	`os.mkdir(path)` `>>> os.mkdir('E://folder')`
دالة rmdir بتمسح الفولدرات هنا مسحنا الفولدر اللى اسمه folder	`os.rmdir(directory_path)` `>>> os.rmdir('E://folder')`
rename بتعمل أعادة تسمية للملف او المجلد. هنا غيرنا اسم الملف test.txt الموجود فى المسار E: إلى test2.txt	`os.rename(old_name',new_name)` `>>> os.rename('E://test.txt', 'E://test2.txt')`
دالة remove بتستخدم لمسح الملفات. هنا مسحنا الملف test2.py	`os.remove(file_path)` `>>> os.remove('E://test2.txt')`

ولو عاوز تعرف أكتر عن موديول os ..ولو عاوز تتعمق أكتر فى نظام الملفات كمل من هنا io.



تطبيق على الملفات

خلاص ننسى os ونرجع للـ files تانى.. شرحتلك طريقة تخزين وأسترجاع الداتا من الملفات بشكل منفصل.. دلوقتى هنستخدمها فى سكريبت تخزين بيانات الطلاب اللى قولنا من مشاكله بنخزن البيانات فى متغيرات فى الميمورى ومحتاجين نخزنها بشكل دائم على الجهاز.

هنعدل فى السكريبت اللى عملناه storages1.py

- فى أول السكريبت هنضيف دالتين.. وحده بتعمل save للـ student dictionary فى ملف نصى.. والتانيه هتعمل read وتقرى البيانات اللى خزناها.

```
def save():
    data=str(students)
    file=open("students.txt","w")
    file.write(data)
    file.close()

def read():
    global students
    try:
      file=open("students.txt","r")
      data=file.read()
      file.close()
      students=eval(data)
    except:
      students={}
```

- دالة save بتحول الـ students dict لنص string وتخزنه فى ملف اسمه students.txt.



- لما الـ dictionary فى البايثون يتحول لـ string هيبقى زى صيغة json string نقدر نقراه تانى من الملف كنص عادي ونحوله إلى dictionary مره تانيه.
- فى دالة read هنقرى الملف students.txt اللى ربما ميكونش موجود ودى أول مره قبل ما تخزن داتا فيه وممكن يحصل exception.. فأنا تفاديت الموضوع ده بجملة try..except.
- هنقرى النص اللى فى students.py واللى هيكون شكله كده مثلاً لو خزنا بيانات طالب اسمه ahmed {'ahmed': {'class': '1', 'section': 'B', 'percent': '90'}} .
- والصيغة دى عبارة عن نص string هى فعلاً على شكل الـ structure بتاع الـ dict ولكنه مش dictionary أقدر اطلع منها values بالـ keys.
- بحل المشكله دى بدالة eval اللى بتعمل execute للنص ده وترجع الـ object المقابل ليه وهو dictionary أقدر اخزنه فى المتغير students.

طيب السؤال هنا وعلى السكريبت القديم.. المفروض نعمل save و read امتى؟ 🤔

- المفروض يا سيدى تعمل save كل مرة تعمل فيها insert لبيانات طلاب جدد أو لما تحذف remove بيانات طالب.. وبالتالى هنعمل call لدالة save فى الدالتين insert و remove.
- والمفروض نعمل read مره وحده بس لما نفتح السكريبت عشان يقرى البيانات المتخزنه فى الملف ويحطها فى الـ students الموجود فى الميمورى.. وبالتالى هنعمل call للدالة read فى السكريبت نفسه.



```
students={}
read()
while True:
     x = input('1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : ')
     if  (x=='1') : insert()
     elif(x=='2') : search()
     elif(x=='3') : remove()
     elif(x=='4') : print(list(students.keys()))
     elif(x=='0') : break
```

ولو كنت توهت ولا حاجة لا سمح اللّه هتحمل السكربت من هنا storages2.py وتشوف

التعديلات اللى عملناها وأبقى ركز شوية والنبى😂

هنجرب نشغل السكربت ونشوف النظام.. المره دى شغلته فى الـ command line.

C:\WINDOWS\py.exe

```
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 1
Student Name : Ahmed
Student Class : 1
Student Section : A
Student Percentage : 88
Any More ? yes
Student Name : Ali
Student Class : 2
Student Section : A
Student Percentage : 87
Any More ? no
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 4
['Ahmed', 'Ali']
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 2
Enter Student Name : Ali
Name : Ali
class : 2
section : A
percent : 87
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? :
```

لو فتحت الملف student.txt اللى هتلاقيه فى نفس المجلد اللى شغلت منه السكربت

هتلاقيه زى كده.



students.txt - Notepad

File Edit Format View Help

```
{'Ahmed': {'class': '1', 'section': 'A', 'percent': '88'}, 'Ali': {'class': '2', 'section': 'A', 'percent': '87'}}
```

لو خرجت من البرنامج وشغلته تانى هتقدر تسترجع البيانات اللى خزنتها..

C:\WINDOWS\py.exe

```
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 4
['Ahmed', 'Ali']
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? : 2
Enter Student Name : Ahmed
Name : Ahmed
class : 1
section : A
percent : 88
1-Insert 2-Search 3-Remove 4-Students 0-Exit ? :
```

نظام الملفات و قواعد البيانات File system vs Database

شوفنا فى الجزء اللى فات طريقة تخزين البيانات فى الملفات files وتعتبر طريقة سهله وسريعة لو عاوز تخزن حاجة وتسترجعها..كل الفكرة إننا بنخزن نص ونسترجع نص لكننا تحايلنا على النص ده بأنه كان على شكل التركيب structure بتاع الـ dictionary وقدرنا بدالة eval نحوله لـ dictionary نقدر نطلع منه keys وvalues وساعدنا على كده سهولة لغة البايثون ولو لغة تانيه كان ممكن تحتاج تعمل parse للمصفوفة students لـ صيغة json ولما تسترجعها تعملها decode هو نفس اللى عملناه بس محسناش بيه عشان اللغة سهله وجميله😂.



كلام جميل..بس لو دققت فى الموضوع أكتر هتلاحظ إن بيانات الطلاب كانت متخزنه فى dict طول الوقت فى الرامة ولما نعمل insert او remove لعنصر واحد بنحول الـ dict كلها لـ str ونخزنها فى الملف النصى .. فلو كان عدد الطلاب كبير كبير هيستهلكوا جزء كبير جداً فى الرامه وعملية الكتابة للملف النصر هتكون بطيئة ومش فعالة.

فظهرت الحاجة لنظام تانى يتكون من جداول.. والجداول دى ليها علاقة ببعضها مش مجرد نص..أقدر اطلع عنصر من الجدول ده بدل ما اطلعه كله فى متغير يستهلك الرامه وأدور جواه..ممكن يكون فيه برنامج تانى او سيرفر تانى هو اللى بيتحكم فى الجدول ملهوش علاقة بالبرنامج بتاعى والبرنامج بتاعى بيكلمه بلغة أستعلامات يقوله خزن القيمة دى فى المكان الفلانى وهات القيمة دى من المكان البتنجانى وهكذا.. 😁

قواعد بيانات SQL Databases

فعملنا نظام قواعد البيانات.. و لغة الأستعلامات دى هى لغة SQL أختصارة لـ Structured Query Language ودى اللغة اللى بتدير برامج وسيرفرات قواعد البيانات.. والبرامج والسيرفرات دى كتير زى Oracle, DB2, and Microsoft SQL Server, MySQL.

يبقى عشان تخزن داتا فى قاعدة بيانات باستخدام لغة البايثون محتاج يكون عندك



1- سيرفر قواعد البيانات Database Server أو البرنامج اللى هيستقبل منك الأوامر والداتا من لغة البايثون ويخزنها وبطريقة منظمة ويسهل عليك حفظها وأسترجعاها..والسيرفرات دى زى MySQL Server أو Microsoft SQL Server.

2- المكتبة الوسيطة بين لغة البايثون وسيرفر قاعدة البيانات Interface Module ودى اللى هتبعت الأستعلامات والداتا بتاعتك للبرنامج بتاع قاعدة البيانات.

قاعدة بيانات SQLite3

كنوع من التبسيط عملوا نوع من قواعد البيانات الخفيفة lightweight ودى تصلح للتطبيقات البسيطة.. مش بتحتاج سيرفر أو برنامج يعمل إدارة للقاعدة serverless يعنى مش هتحمل أى برامج عشان تستخدمها.. يتم إدارتها من خلال مكتبة بلغة الـ C تابعة للغة ويتم حفظ ملف قاعدة البيانات فى مسار بيحدده المستخدم.

لو عاوز تعرف عنها أكتر من هنا SQLite .

والـ standard library للغة البايثون فيها فيها موديول sqlite3 ده الإنترفيس بتاع القاعدة اللى هنتعامل معاها من خلاله.. والجميل إن قواعد بيانات SQLite يتم التعامل معاها باستعلامات SQL كأى Database Server زى Oracle و Mysql

موديول sqlite3

- فى الأول عملنا import للمكتبه.. وهنا اخترت اسم sql للتبسيط.

```
>>> import sqlite3 as sql
```



- بعدين هنعمل connection object بدالة connect اللى بتاخد برامتر وهو الاسم والمسار اللى هتتحفظ بيهم قاعدة البيانات

```
>>> con=sql.connect("E://database.db")
```



- لو فتحت المسار ده واللى أنا اخترته عندى فى البارتيشن :E كالعاده وانت طبعاً حر فى مساراتك 😂 هتلاقى البرنامج أنشا ملف اسمه database.db.
- واحد هيسألنى يعنى بعد اللفة الحوارات دى هيخزن الداتا فى قاعدة البيانات اللى هى ملف أصلاً ؟.. نعم بس هنا هتتعامل مع الملف بلغة أستعلامات محترمة وبطريقة منظمة أصبر وهتشوف 😂
- الـ connection object اللى عملناه con فيه مجموعة methods هنستخدمهم cursor,commit,close

الـ cursor

تانى خطوة بعد ما عملنا connection object وده اللى بينشىء قاعدة البيانات هنعمل cursor object بدالة cursor وده اللى فيه الدوال اللى هتكلم قاعدة البيانات وتبعتلها الأستعلامات وتخزن وتسترجع القيم فيها.

```
>>> import sqlite3 as sql
>>> con=sql.connect("E://database.db")
>>> cur=con.cursor()
```



الـ cursor object فيه مجموعة دوال methods هنستخدم شوية قليلين منهم زى execute,fetchone,fetchall,rowcount وهنعرف فايدتهم ايه بس الموضوع كبير طبعاً لازم تكمل مع نفسك.

أستعلامات SQL quesries

عشان تقدر تتعامل مع أى sql server لازم تتعلم لغة الأستعلامات واللغة دى عبارة عن أكواد بتنشئ الجداول فى قاعدة البيانات بأسماء وخانات أحنا بنحددها...وبتعمل أدخال لقيم فى الجداول..وتعمل تحديث للقيم دى فى الجدال لو عاوزين نغيرها..وبتعمل حذف للصف فى الجدول أو تحذف الجدول كله وهكذا.

لو معنكدش فكرة عن لغة sql هناخد فكرة عن بعض الأستعلامات وهشرحها على السريع.. وللمزيد برشحلك الـ tutorial دى sql على موقع w3schools

إنشاء جدول CREATE TABLE

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
)
```

- عشان تنشئ جدول فى قاعدة البيانات بتسخدم جملة CREATE TABLE بعدها اسم الجدول..وبين القوسين اسم الأعمد فى الجدول ونوع الداتا اللى هنخزنها فيهم.



- أنواع البينات ممكن تكون INT,FLOAT لو نص هيبقى TEXT أو MEDIUMTEXT أو LONGTEXT .. وبتحدد النوع حسب طول البيانات اللى هتخزنها هل هى اسم رسالة,تعليق فتخزنهم فى text ولا ملف كبير,كتاب فتروح للـ MEDIUMTEXT أو LONGTEXT وهكذا.
- فيه نوع اسمه VARCHAR للنصوص الصغيرة وبتقدر تحدد طول النص أو عدد الحروف اللى بتخزنها فيه بالشكل ده VARCHAR(length)
- الفرق بين TEXTوVARCHAR الاول إن فى حالة VARCHARبتحدد طول النص ولو النص كان أطول من عدد الحروف المحدد فى الحالة دى هيقطعه ومش هيخزن غير الطول المسموح..أما text الطول مفتوح.

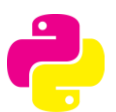
الفرق التانى إن الداتا من نوع TEXT بتتخزن فى ملفات خارجية عن ملف جدول قاعدة البيانات والكلام ده في بعض الـ Database Servers .

- فلو عندك نصوص قصيرة زى اسم بريد الكترونى فصل دراسى تستخدم VARCHAR ولو استخدمت TEXT لكله عادى برده مفيش مشاكل.

شوف المثال ده..

```
CREATE TABLE user(
      id int,
     age  int,
    name text,
    email text,
)
```

- الأستعلام ده هينشىء جدول اسمه user فيه 4 أعمدة id,age,name,email.



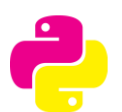
id	age	name	Email

- المفروض تكون فهمت إن عملية تخزين الداتا فى جدول قاعدة البيانات بتكون على شكل إضافة صفوف Rows كل صف فيه قيم لكل cell أو عمود.
- خانة id يفضل انها تحمل رقم unique يبقى مختلف فى كل صف.
- كما هو شائع بيخلوا الخانة id رقم unique وبيزيد تلقائى كل ما تعمل insert لداتا جديدة فى قاعدة البيانات.. وتقدر تعمله بأنك تستعمل INTEGER PRIMARY KEY بدل من كلمة int.

و ممكن نستخدم أسلوب الشرط فى أستعلام CREATE وخليه بالشكل ده

```
CREATE TABLE IF NOT EXISTS table_name (
    column1 datatype,
)
```

- فى الحالة دى هيعمل جدول جديد بالاسم ده لو مكنش موجود.
- لو عملت الأمر ده CREATE table_name وكان الجدول موجود ومتخزن فيه قيم هيمسح الجدول الموجوده وينشئه من جديد.
- يبقى بنستخدم الأستعلام IF NOT EXISTS عشان ننشىء الجدول فى أول مرة نشغل فيها التطبيق غير كده مش محتاجين نعمله CREATE تانى.



مسح الجدول DROP TABLE

عشان تمسح الجدول يتم بجملة DROP

```
DROB TABLE  table_name
```

إدراج الصفوف INSERT INTO

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...)
```

- عشان تخزن داتا فى الجدونل بتستخدم جملة INSERT INTO بعدها اسم الجدول table_name وبين القوسين اسماء الأعمده اللى هتدخل فيها الداتا وبعدين جملة VALUES وبعدها القيم اللى هتدخلها فى الأعمده بالترتيب اللى حددته فى الأقواس وعاملك الوان أهو عشان تعرف إن كل قيمة هتروح للـ cell المقابلة فى الجدول.

```
INSERT INTO user (age, name,email) VALUES
(20,'mahmoud','user@email.com')
```

ده مثال لحفظ بيانات فى الخانات age,name,email فى الجدول.

- لاحظ إننا معملناش أدخال لقيمة فى خانة id لأنها هتاخد قيمة تزايدية بشكل تلقائى بسبب الأمر ده INTEGER PRIMARY KEY
- القيم غير الرقمية لازم تتحط بين quotes .. شكل الجدول هيكون كده

id	age	name	Email
1	20	ahmed	ahmed@email.com



تحديث الصفوف UPDATE

عشان تعمل تحديث لقيمة فى الجدول بتتم بالشكل ده..

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- بتستخدم جملة UPDATE بعدها اسم الجدول.. بعدين جملة SET وبعدها كل عمود فى الصف والقيمة الجديدة وأخير حاجة وأهم حاجة الـcondition وده اللى وده بيحدد أنت عاوز تحدث القيم دى فى أي صف.

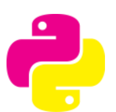
```
UPDATE user SET name ='python' WHERE id=1;
```

- مثلاً هنا بقوله فى الجدول user شوف الخانة اللى فيها الـ id يساوى 1 وغير القيمة اللى فى العمود name وخليها python.

حذف الصفوف DELETE

```
DELETE FROM table_name WHERE condition;
```

```
DELETE FROM user WHERE name='python';
```



- هنا بقوله أحذف الصف أو الصفوف الموجودة فى الجدول user اللى فيها القيمة فى العمود name تساوى python.
- عشان تحذف كل الصفوف فى الجدول بتستخدم الاستعلام ده.

```
DELETE * FROM table_name;
```

إسترجاع القيم SELECT

عشان تسترجع القيم من الجدول ده بيتم بجملة SELECT

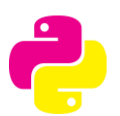
```
SELECT column1, column2, ..FROM table_name;
SELECT * FROM table_name;
```

مثلاً لو عندنا جدول زى كده وأستخدمنا الإستعلامات اللى تحت..

id	name	Age
1	mahmoud	23
2	ali	22
3	python	30

```
SELECT name FROM table_name;
```

- هيرجع حاجة اسمها result set فيه كل القيم اللى في العمود name وهو mahmoud,ali, python



```
SELECT  * FROM table_name WHERE  id=1
```

- بالأستعلام ده هيروح للصف اللى فيه الـ id =1 ويرجع الصف كله .id=1,name=mahmoud,age=20

```
SELECT * FROM table_name
```

- هيرجع الجدول كله.

وبالأوامر أو الإستعلامات دى نكون أتعلمنا ازاى نكريت جدول بالخانات وأنواع البيانات اللى عايزينها وعرفنا ازاى نمسح الجدول ونخزن فيه ونسترجع ونحدث القيم.. تعالى بقى نرجع نتعامل بلغة البايثون.

دالة execute

الحكاية وسعت مننا وتوهنا ونسينا كنا بنتكلم عن ايه 😁 أفكرك احنا عملنا قاعدة بيانات وعاوزين نكلمها بالـ SQL queries اللى كنت بعلمك فيها دلوقتي وعملنا connection object اللى أنشأ قاعدة البيانات نفسها ومنه عملنا cursor object وقولنا فيه الدوال اللى بتبعت الـqueries لقاعدة البيانات زى دالة execute .

```
cur.execute('SQL STRING')
```

دالة execute بتاخد برامتر واحد وهو كود أو أستعلام SQL.



دالة commit

دى تابعة لـلـ connection Object و بتعملها call بعد ما تخلص الأستعلامات وبدونها مفيش حاجة هتتحفظ فى قاعدة البيانات كأنك معملتش حاجة.

```
cur.execute('SQL STRING')
con.commit()
```

دالة close

أخيراً دى بنستخدمها فى نهاية الكود عشان نقفل الإتصال connection مع قاعدة البيانات..وبديهى إنها تكون تابعة لـلـ connection object.

```
con.close()
```

- هنعمل قاعدة بيانات باسم database.db ونعمل منها cursor object اسمه cur

```
>>> import sqlite3 as sql
>>> con=sql.connect("E://database.db")
>>> cur=con.cursor()
```

- هنعمل جدول اسمه user فيه الخانات id,name,age .

```
>>> cur.execute("CREATE TABLE user(id INTEGER PRIMARY KEY,name
VARCHAR(10),age INTEGER)")
```

- هنعمل insert لصفين فى الجدول فى الخانات name,age وبعدين نعمل commit عشان القيم تتحفظ.



```
>>> cur.execute("INSERT INTO user(name,age)
VALUES('Mahmoud','23')")
>>> cur.execute("INSERT INTO user(name,age)
VALUES('Ahmed','22')")
>>> con.commit()
```

- هنسترجع كل القيم من الجدول بإستعلام * SELECT فى متغير اسمه table وعن طريق دالة fetachall بترجع list فيها كل صف فى الجدول على شكل tuple.

```
>>> table=cur.execute('SELECT * FROM user')
>>> table.fetchall()
[(1, 'Mahmoud', 23), (2, 'Ahmed', 22)]
>>>
```

- دى طرق مختلفة لقراءة صف بـ id محدد او قراءة العمود name كله أو cell وحده.

```
>>> cur.execute("SELECT * FROM user WHERE id=1").fetchall()
[(1, 'Mahmoud', 23)]
>>> cur.execute("SELECT name FROM user").fetchall()
[('Mahmoud',), ('Ahmed',)]
>>> cur.execute("SELECT name FROM user WHERE
id=2").fetchall()
[('Ahmed',)]
```

- تحديث خانة أو صف فى الجدول بجملة UPDATE

```
>>> cur.execute('SELECT * FROM user').fetchall()
[(1, 'Mahmoud', 23), (2, 'Ahmed', 22)]
>>> cur.execute("UPDATE user SET name='python' WHERE id=2")
>>> con.commit()
>>> cur.execute("SELECT * FROM user").fetchall()
[(1, 'Mahmoud', 23), (2, 'python', 22)]
```



تطبيق على قاعدة بيانات SQLite

لازلنا فى فصل طرق التخزين.. أتكلمنا عن التخزين فى الملفات الـ File System واتكلمنا على التخزين فى قواعد بيانات SQL .

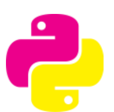
هنعدل على سكريت تخزين بيانات الطلاب storages1.py ونستخدم فيه قاعدة بيانات SQLite بدل الملف النصى.

- فى البداية هنضيف موديول sqlite3 ونعمل connection object و cursor object وننشىء جدول students فى حالة إنه مكنش موجود(عند اول مره لتشغيل السكريت)

```
import sqlite3 as sql
con=sql.connect("database.db")
cur=con.cursor()
query="""CREATE TABLE IF NOT EXISTS students(
            name VARCHAR(20),
            class VARCHAR(5),
            section VARCHAR(5),
            percent int
        )
"""
cur.execute(query)
```

- هنغير فى دالة insert() عشان نخليها تعمل insert لبيانات الـ students فى قاعدة البيانات.

```
not_found=cur.execute("SELECT name FROM students WHERE
name='%s'"%name).fetchall() == []
if not_found:
     query="""INSERT INTO students(name,class,section,percent)
               VALUES('%s','%s','%s','%s')"""%(name,_class,section,percent)
else:
    query="""UPDATE students SET class='%s',section='%s',percent='%s'
```



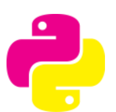
```
            WHERE name='%s' """%(_class,section,percent,name)

cur.execute(query)
con.commit()
```

- في البداية وكالعاده داله insert هتقرى من المستخدم name,class,section,percent مفيش أختلاف
- طيب .. لما نعمل أدخال لمستخدم جديد ممكن يكون اسمه موجود بالفعل وبالتالى لو أستخدمنا جملة INSERT INTO هتلاقيه عمل صف بنفس الاسم تانى وده احنا مش عاوزينه يحصل..اللى عاوزينه لو دخلنا بيانات طالب ولقى الاسم موجود فى الحالة دى يعمل UPDATE للصف مش INSERT.
- فى الأول بنتأكد إذا كان الاسم موجود باننا نعمل SELECT للعمود اللى اسمه name فى الجدول students و الداله setchall قولنا بترجع العمود ده فى List .. لو كان الاسم مش موجود مسبقاً هترجع لستة فاضيه [ ] يعنى و not_found قيمته هتبقى True لو الاسم موجود not_found هيبقى False.
- فى حالة الاسم كان مش موجود not_found=True هنفذ الأستعلام بتاع INSERT
- لو الاسم كان موجود هنفذ الإستعلام بتاع UPDATE وأخليه يحدث بياناته.
- ومننساش نعمل commit فى الأخر عشان يحفظ القيمة فى قاعدة البيانات.

داله search عملنا فيها تعديلات بسيطة.

```
def search():
    name=input("Enter Student Name : ")
    rows=cur.execute("""SELECT * FROM students
    WHERE name='%s'"""%name).fetchall()
```



```
    for row in rows:
        print('Name :',row[0])
        print('Class :',row[1])
        print('Section :',row[2])
        print('Percent :',row[3])
        return
    print('student not found')
```

- فى الاول المستخدم هيدخل الاسم name اللى عاوز يطلع بياناته وبعدين هنعمل SELECT لكل الصفوف فى الجدول students اللى فيها name يساوى الاسم اللى دخله المستخدم.
- و دالة setchall كالعادة بترجع لـ list بكل الصفوف المطابقة للشرط إن وجدت وتخزنها فى المتغير rows.
- بعدين بنعمل iteration على المتغير rows اللى بيرجع tuple اسمه row بالشكل ده مثلاً (Mahmoud,1,A,90) بنطلع العناصر بالـ index وبنطبعها..

حذف بيانات الطلاب..دالة remove

```
def remove():
    name=input("Enter Student Name : ")
    result=cur.execute("DELETE FROM students WHERE name='%s'"%name)
    con.commit()
```

- المستخدم بيدخل اسم الطالب اللى عاوز يحذفه name ويتم تنفيذ query جملة DELETE FROM اسم الجدول students والشرط هو إنه يمسح الصف أو الصفوف اللى فيها name هيطابق الاسم اللى دخله المستخدم..

عرض أسماء كل الطلاب



```
def students():
    col=cur.execute("SELECT name FROM students").fetchall()
    for name in col:print(name[0])
```

- عشان نعرض أسامي قبل كده كنا بنطلع الـ keys في الـ students dictionary.

أما دلوقتي محتاجين نعمل SELECT للعمود اللي اسمه name في جدول students

و دالة fethall بترجع العمود على شكل list وكل اسم جوه tuple بالشكل ده

[('Ahmed',), ('Mahmoud',)].. بنعمل iteration عليه ونطبعه.

ولو عاوز تحمل الكود النهائي للسكريبت فمن هنا storages3.py



الفصل 5 – بعض المكتبات المشهورة

الفهرس



فى الفصل ده هنتكلم عن بعض المكتبات المشهوره للغة البايثون واللى هنقدر نستخدمها فى تطبيقات كتير زي تحميل الفيديوهات من الأنترنت أو معالجة الصور وتحرير الفيديوهات. ومش لازم تشتغل على مكتبات الشابتر ده كلهم ..بس بضمنلك إنك هتلاقى حاجة شيقة وهتستفاد بيها جداً..

مكتبة youtube-dl

لو مكنتش سمعت عنها قبل كده فأعتقد من الاسم قدرت تعرف انها مكتبه ليها علاقة باليوتيوب او بتحمل الفيديوهات من اليوتيوب.

مكتبة youtube-dl من المكتبات العظيمة اللى بتساعدك إنك تحمل الفيديوهات مش من على اليوتيوب بس ولكن من أغلب مواقع تحميل الفيديوهات المشهوره وحتى الفيس بوك.

ده موقع المكتبة على GitHub youtube-dl واللى فيه documentation بتاعتها والمواقع اللى بتدعم التحميل من عليها.

youtube-dl — Download videos from YouTube (and more sites)

youtube-dl is a command-line program to download videos from YouTube.com and a few more sites. It requires the Python interpreter (2.6, 2.7, or 3.2+), and it is not platform specific. We also provide a Windows executable that includes Python. youtube-dl should work in your Unix box, in Windows or in Mac OS X. It is released to the public domain, which means you can modify it, redistribute it or use it however you like.

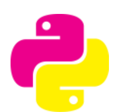
Documentation

Download

Support

Develop

About



تثبيت المكتبة

مكتبة youtub-dl مكتوبة بلغة البايثون وتقدر تثبتها عن طريق pip.

هتفتح الـ command line وتثبتها باستخدام pip ..

```
pip install youtube-dl
```

الاستخدام

لو كنت فتحت موقع المكتبة هتلاقى المكتبة متوفرة لنظام ويندوز على شكل ملف تنفيذى exe. تقدر تستخدمه من خلال الـ command line من غير ما تتحتاج لغة البايثون ولا تكتب اكواد.

ومتوفرة على شكل Python Package ودى اللى ثبتناها باستخدام pip ونقدر نستخدمها عن طريق لغة البايثون أو من الـ command line برده.

تحميل فيديو من خلال الـ command line

- أفتح الـ command line أياً كان نظام تشغيلك وأكتب فيه youtbe-dl

```
C:\Users\Mahmoud>youtube-dl
Usage: youtube-dl [OPTIONS] URL [URL...]

youtube-dl: error: You must provide at least one URL.
Type youtube-dl --help to see a list of all options.
```

لو ظهرت الرسالة دى يبقى المكتبة أتثبتت وتمام .



- عندم وتقدر تحمل بيها الفيديوهات من خلال الـ command line.

```
youtube-dl video_url
```

هتكتب youtube-dl + عنوان الفيديو من اليوتيوب/فيس بوك/ أى موقع بتدعمه المكتبه وبالطريقة دى هتحملك الفيديو بأفضل جودة متوفرة.

ده اللى بيظهر قدامك وأنت وبتحمل فيديو خلال الـ CLI.

```
C:\Users\Mahmoud>youtube-dl https://www.youtube.com/watch?v=SgNZYC4khAo
[youtube] SgNZYC4khAo: Downloading webpage
[youtube] SgNZYC4khAo: Downloading video info webpage
[youtube] SgNZYC4khAo: Downloading js player vflvABTsY
[youtube] SgNZYC4khAo: Downloading js player vflvABTsY
WARNING: Requested formats are incompatible for merge and will be merged into mkv.
[download] Destination: ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯   ▯▯▯▯▯ ▯▯▯▯▯▯▯-SgNZYC4khAo.f248.webm
[download]   1.0% of 27.07MiB at 58.14KiB/s ETA 07:51
```

لو ظهرت مشاكل فى التحميل على أى موقع وكنت مثبت المكتبة من فتره يبقى لازم تعملها تحديث upgrade باستخدام pip.

```
pip install youtube-dl --upgrade
```

والموضوع ده مهم عشان المكتبه بتتحدث كل أيام قليلة عشان تقدر تطلع روابط التحميل من المواقع المختلفه واللى بتغير طرق تشفيرها وأخفاءها كل فتره.. وتقدر تتابع وتشوف أخر release بتنزل من هنا download.



المكتبة فيها خيارات مهمة تقدر تخليك تحمل الفيديوهات بصيغ وجودات مختلفة وبتقدر تحمل playlist أو channel كاملة من اليوتيوب وأكتر من كده..مش عاوز اتوسع قوى فى الموضوع بس بنصحك لازم تشوف الـ documentation على موقع GitHub.

تحديد الجودة FORMAT SELECTION

Youtube-dl بشكل أفتراضى بيحمل أفضل صيغة متوفرة للفيديو.. و عشان تحدد جودة الفيديوهات اللى هتحملها, المكتبة بتوفر أكتر من طريقة زى كده

```
youtube-dl -f video_format video_url..
```

والـ video format أو video quality بتكون كده

Video description	Video exetention
best worst bestvideo worstvideo bestaudio worstaudio	3gp, aac, flv, m4a, mp3, mp4, ogg, wav, webm



ده مثال لتحميل فيديو بأفضل جودة متوفرة للصوت bestaudio أو الأمر التانى بيحمل الفيديو بصيغة mp4.

```
youtube-dl -f bestaudio  video_url
youtube-dl -f mp4  video_url
```

عشان تعرف كل الصيغ المتاحة بتكتب الامر F- قبل الرابط ولاحظ إن الـ F كابتال.

```
youtube-dl -F video_format video_url..
```

```
C:\Users\Mahmoud>youtube-dl -F https://www.youtube.com/watch?v=1D93k0zh8uA
[youtube] 1D93k0zh8uA: Downloading webpage
[youtube] 1D93k0zh8uA: Downloading video info webpage
[info] Available formats for 1D93k0zh8uA:
format code  extension  resolution note
249          webm       audio only DASH audio   74k , opus @ 50k, 11.99MiB
171          webm       audio only DASH audio   86k , vorbis@128k, 14.80MiB
250          webm       audio only DASH audio   90k , opus @ 70k, 14.56MiB
140          m4a        audio only DASH audio  129k , m4a_dash container, mp4a.40.2@128k, 24.78MiB
251          webm       audio only DASH audio  147k , opus @160k, 24.41MiB
160          mp4        256x144    144p   29k , avc1.4d400c, 25fps, video only, 3.03MiB
133          mp4        426x240    240p   48k , avc1.4d4015, 25fps, video only, 6.12MiB
278          webm       256x144    144p   72k , webm container, vp9, 25fps, video only, 8.58MiB
```

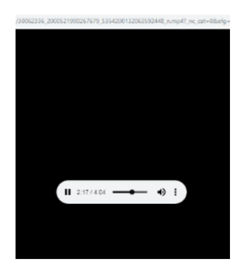
- ومن الأوامر المهمة get-url– ده بيطلعلك رابط التحميل المباشر للفيديو لو عاوز تحمله ببرنامج تحميل تانى مثلاً او تحطه فى video player فى موقع معين.

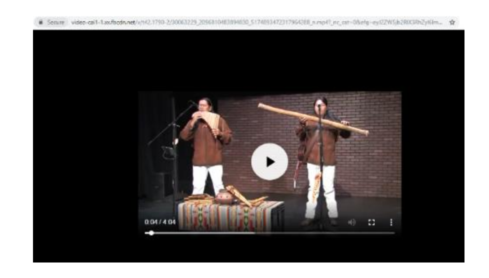
```
youtube-dl --get-url video_url
```

```
C:\Users\Mahmoud>youtube-dl --get-url https://www.facebook.com/1541017819491365/videos/2002320483361094/
https://video-cai1-1.xx.fbcdn.net/v/t42.1790-2/30063229_2096810483894830_5174893472317964288_n.mp4?_nc_cat=0&efg=eyJ2ZW
jb2RlX3RhZyI6ImRhc2hfdjRfcGFzc3Rocm91Z2hfZnJhZ18yX3ZpZGVvIn0=&oh=91f03f51990de904d7be205401abaf6f&oe=5B946EF2
https://video-cai1-1.xx.fbcdn.net/v/t42.1790-2/30062336_2000521990267679_5354200132063592448_n.mp4?_nc_cat=0&efg=eyJ2ZW
jb2RlX3RhZyI6ImRhc2hfdjRfcGFzc3Rocm91Z2hfZnJhZ18yX2F1ZGlvIn0=&oh=c47630136696d9f8d27826da3d8f1b1c&oe=5B94700E
```

فى الصورد دى حاولت أجيب رابط التحميل المباشر لفيديو على الفيس بوك راح البرنامج طلع اتنين url الأول فيديو والتانى الملف الصوتى.







استخدام مكتبة youtube-dl مع البايثون

من بدرى بنتكلم عن طريقة استخدام youtub-dl فى الـ command line مع إن الكتاب

أصلاً بيشرح لغة البايثون والمكتبة فى الأصل ثبتناها على اساس أنها Python Package

فتعالوا نشتغل بايثون شوية.

ده الشرح فى الوثيقة لاستخدام youtube-dl مع لغة البايثون embedding-youtube-dl

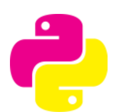
```
import youtube_dl

ydl_opts = {}
ydl=youtube_dl.YoutubeDL(ydl_opts)

url=input('Paste Video URL : ')
ydl.download([url])
```

Download script youtube-dl.py from GitHub

- متغير ydl_opts دى بيحدد إعدادت الفيديو اللى عايزين نحمله زى الصيغة والجوده.
  و هنا عملناه dict فاضى وبالتالى هيحمل أفضل جوة متوفرة
- داله download بتاخد الـ urls كـ list مش str عشان لو عاوز تحمل أكتر من فيديو.
- هندخل الرابط للسكريبت عن طريق داله input.



- وده شكل السكريبت لما تشغل

```
Paste Video URL : https://www.youtube.com/watch?v=hzVHVGClgIw
[youtube] hzVHVGClgIw: Downloading webpage
[youtube] hzVHVGClgIw: Downloading video info webpage
WARNING: Requested formats are incompatible for merge and will be merged i
[download] Resuming download at byte 671776
[download] Destination: ▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯ ▯▯▯▯ ▯▯▯▯▯▯▯▯ ▯▯▯▯▯ ▯▯▯▯▯▯▯
[download]   2.6% of 105.63MiB at 203.92KiB/s ETA 08:36
```

- عشان تحدد جودة التحميل ده بيتم من خلال المتغير ydl_opts ودى طرق تحديد أكتر من جودة للفيديو

```
ydl_opts = {'format': 'bestvideo'}
ydl_opts = {'format': 'bestaudio'}
ydl_opts = {'format': 'mp4'}
```

طبعاً أنت كمستخدم عادى تقدر تشتغل على command line حتى من غير ما تكتب أكواد بايثون ..تلجأ بس للبايثون لو عاوز تعمل تطبيق أو موقع يستخدم المكتبة عشان يحمل الفيديوهات والمستخدم يختار الصيغة اللى عاوزها.

لو عاوز أمثلة أكتر عن استخدام المكتبة بلغة البايثون تقدر تشوف الموقع ده جايب كمية أمثلة كبيرة youtube_dl.



وقبل ما اخلص كلام عن youtbe-dl هستغل الموقف وأوريكم برنامج youtuber ده برنامج عملته بلغتة الـ #C (مش البايثون 😂) مبني على الـ windows executable اللي بتنزل مع المكتبة وعاملها واجهة رسومية ودي للناس الغلابه اللي مبتعرفش تشتغل على الـ command line.

youtuber
×
start add
YouTube
video playlist quality best 3gp - 176x144
video quality state

start add
YouTube
video playlist quality best 3gp - 176x144
video quality state
https://www.youtube.com/watch?v=xMYkqR8Vifw best waiting
change quality
remove video
clear list
best
best video
worst video
best audio
worst audio
3gp - 176x144
3gp - 320x240
mp4 - 480x360
webm - 640x360
mp4 - 960x720

مكتبات ffmpeg

من المنصات العظيمة والمشهوره لتحرير الفيديوهات.. وهو عباره برنامج بيشتغل على الـ command line وبيستخدم لتحويل صيغ الفيديوهات والملفات الصوتيه والصور المتحركة والقص والتقطيع والتسجيل وغيره.. ffmpeg.

برنامج ffmpeg بينزل على شكل عن ملف تنفيذى executable file أو binary file بالنسبة لبتوع لينكس.. بنتعامل معاه من خلال command line. ومش زى مكتبة youtube-dl اللى أصلاً مكتوبة بلغة البايثون.

بقولك كده ليه ؟.. عشان تعرف أن البرامج اللى بتبقى موجودة بالشكل ده لما نحاول نتعامل معاها بلغة البايثون يتطلب وجود حاجتين : البرنامج نفسه اللى بيشتغل بس على الـ command line زى ffmpeg والمكتبة اللى بتكلم البرنامج ده من لغة البايثون وبتسمى wrapper.

وأنا وبعمل الكتاب ده لقيت أكتر من wrapper ممكن تشتغل عليه زى مكتبة ffmpy ومكتبة ffmpeg-python .. أنا قررت اجرب الأخيره وممكن متكونش الأفضل بس هنجربها ونشوف.



مكتبة ffmpeg-python

هتثبت المكتبة عن طريق pip .. ولو عاوز تعرف أكتر عن المكتبة وتشوف الوثيقة بتاعتها من هنا ffmpeg-python.

```
pip install ffmpeg-python
```

برنامج ffmpeg

تقدر تحمل الـ builds المناسبة لنظام تشغيلك وخصوصاً لويندوز من هنا مش من الموقع الرسمي ffmpeg.zeranoe.com

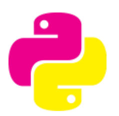
و لو انت زيي وشغال على ويندوز هتفك ضغط الملفات في أي مكان تحبه. أنا هحطهم في البارتيشين C جوه الفولدر هتلاقي فولدر اسمه bin جوه الملفات ffmpeg,ffplay,ffprobe

ffmpeg.exe
ffplay.exe
ffprobe.exe

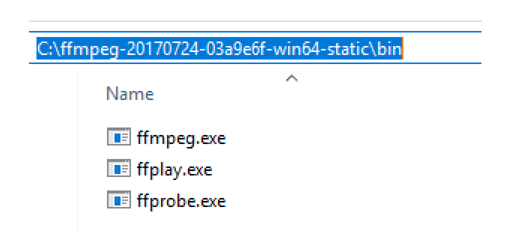
- المفروض نستخدم ffmpeg من خلال الـ command line فأنت هتفتح الـ cmd وتكتب ffmpeg

```
C:\Users\Mahmoud>ffmpeg
'ffmpeg' is not recognized as an internal or external command,
operable program or batch file.
```

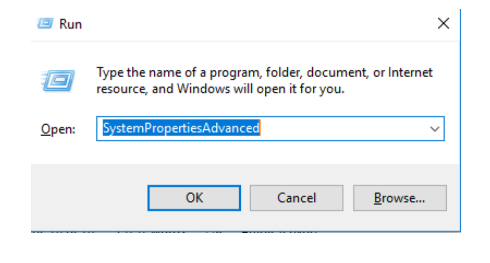
- بيقولك إنه مش عارف مين ffmpeg .. ولسه فاضل خطوه إنك تحط مسار الفولدر bin اللي فيه ملف ffmpeg في الـ environment variables عشان يقدر يشوفه الـ command line من أي مكان.



أيا كان المكان اللى حطيت فيه هتروح تنسخ العنوان وتحطه فى المسار PATH فى الـenvironment varaibles.



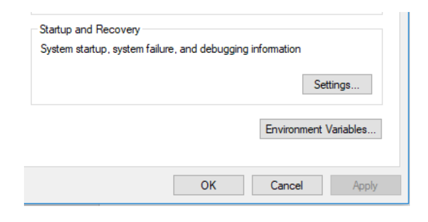
لو مش عارف إزاى تجيب الـ environment variables أفتح الشاشة بتاعة run او دوس على window+R وأكتب فيها SystemPropertiesAdvanced

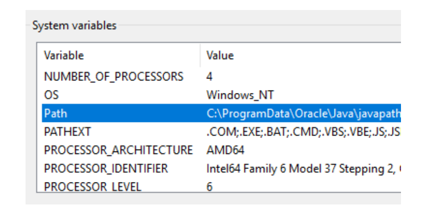


أو أفتح الـ cmd وأكتب فيه start SystemPropertiesAdvanced

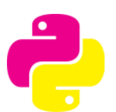
```
C:\Users\Mahmoud>start SystemPropertiesAdvanced
```

بعد كده دوس على زرار environment variables وعدل على Path وضيف فيه المسار اللى فيه ffmpeg.





أحنا كل ده بنحاول نحط المسار بتاع الفولدر bin اللى فيه ملف ffmpeg.exe فى المسارات path فى الـ environment variables فحاول تتشقلب وتضيفه بأى طريقة 😂.



طبعاً أخواتنا بتوع لينكس عنهم Package Manager بيثبت البرنامج بعيداً عن الحوارات دى.

- دلوقتى أفتح الـ command Line واكتب فيه ffmpeg ولو كنت عملت الخطوات اللى فاتت صح هيرد عليك ..

```
C:\Users\Mahmoud>ffmpeg
ffmpeg version N-86848-g03a9e6f Copyright (c) 2000-2017 the FFmpeg developers
  built with gcc 7.1.0 (GCC)
```

استخدام ffmpeg فى الـ command line.

فى البداية هنجرب ffmpeg على command line وقولنا أنه بيستخدم لتحويل صيغ الفيديو والصوت والتقطيع والقص وحاجات كتير جداً بتلاقيها فى أى برنامج video editor.

تحويل صيغ الفيديو video converstion.

طريقة التحويل بين صيغ الفيديوهات بتتم بالشكل ده.

```
ffmpeg -i input output
```

- فى المثال اللى تحتك بنحول فيديو input.mp4 إلى صورة متحركة output.gif.

```
ffmpeg -i input.mp4 output.gif
```

طبعاً ffmpeg مش مجرد برنامج لتحويل الصيغ بس. تقدر من خلاله تتحكم فى جودة وأبعاد وسرعة الفيديو او الصوت او الصورة المتحركة اللى شغال عليها فلو مهتم وحبيت الموضوع تقدر تعرف أكتر من هنا documentation.



هناخد مثال بسيط على استخدام برنامج ffmpeg في الـ command line.



ده فيديو موجود على GitHub عاوزين نحوله لصورة متحركة https://ma7moud3ly.github.io/video.mp4

مش لازم تحمل الفيديو ممكن تستخدمه وهو اونلاين.

- هتفتح الـ command line وتوجهه للمكان اللى عاوز تحفظ فيه الصورة الناتجة
- هتنسخ رابط الفيديو وهتنفذ الأمر ده

```
C:\Users\Mahmoud>E:
E:\>ffmpeg -i https://ma7moud3ly.github.io/video.mp4 image.gif
```

هتلاقى الفيديو أتحول لصورة متحركة image.gif ومحفوظة فى المسار اللى فتحت عليه.

استخدام مكتبة ffmpeg-python

تعالوا نعمل الكلام ده بلغة البايثون.. بس فى الأول لازم تكون مثبت برنامج ffmpeg وضايفة للـ environment path ومثبت مكتبة ffmpeg-python عن طريق pip زى ما عملنا.

التحويل من صيغة لأخر

```
import ffmpeg
input = ffmpeg.input('https://ma7moud3ly.github.io/video.mp4')
output = ffmpeg.output(input, 'image.gif')
ffmpeg.run(output)
```

Download script ffmpeg1.py from GitHub



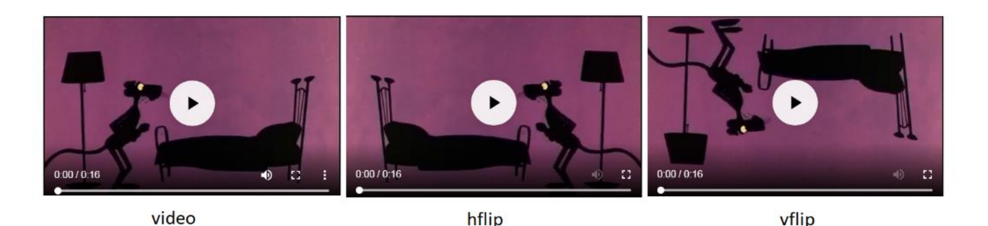
- دالة input بتاخد عنوان الفيديو اللى عاوز تحوله وبترجع object فيه مجموعة من الدوال هنتكلم عن بعضهم فى الأمثله اللى جاية
- دالة output بتاخد برامترين.. الـ input object والاسم والصيغة اللى عاوز تحفظ الفيديو بيها.
- أخيراً دالة run بتنفذ الأوامر دى فى برنامج ffmpeg.

عمل flip للفيديو

عشان تعمل أنعكاس للفيديو filp بشكل أفقى او رأسى بتستخدم دوال hflip و vflip.

```
import ffmpeg
input = ffmpeg.input('https://ma7moud3ly.github.io/video.mp4')
hflip=input.hflip() ; vflip=input.vflip()
ffmpeg.run(ffmpeg.output(hflip, 'hflip.mp4'))
ffmpeg.run(ffmpeg.output(vflip, 'vflip.mp4'))
```

Download script ffmpeg2.py from GitHub



video hflip vflip



قص الفيديو crop

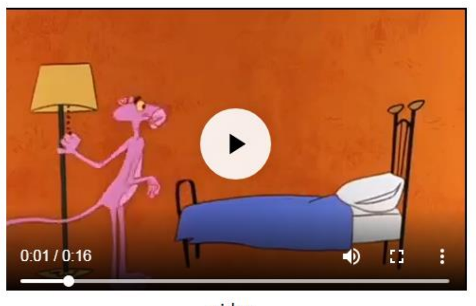
بتستخدم دالة crop اللى بتاخد 4 برامترات.. x,y وده بعد الجزء اللى هتقطعه من أعلى يسار الفيديو.. وبعدين width,height ودول الطول والعرض للمربع اللى هتقطعه.

```
import ffmpeg
input = ffmpeg.input('https://ma7moud3ly.github.io/video.mp4')
crop=input.crop(x=0,y=0,width=200,height=200)
ffmpeg.run(ffmpeg.output(crop, 'crop.mp4'))
```

Download script ffmpeg3.py from GitHub



crop



video

الكتابة على الفيديو

لكتابة نص على الفيديو بنستخدم دالة drawtext زى المثال ده.

```
import ffmpeg
input = ffmpeg.input('https://ma7moud3ly.github.io/video.mp4')

text=input.drawtext(text='Pink Python ^_^',
fontcolor='red',
fontsize=30,
box=1,
boxcolor='black@0.5',
x='(w-text_w)/2',
```



```
y='(h-text_h)/2'
)

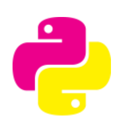
text=text.drawtext(text='By Mahmoud Aly',
fontcolor='white',
fontsize=15,
x='(w-text_w)/2',
y='((h-text_h)/2)+30'
)

output=ffmpeg.output(text, 'text.mp4')
ffmpeg.run(output)
```

Download script ffmpeg4.py from GitHub

- الدالة drawtext بتحدد فيها النص اللى عاوز تكتبه وخصائص النص زى fontcolor,fontsize والأحداثيات x,y ولو عاوز box أو خلفية للنص وهكذا..





مكتبة OpenCV

شيء غير منصف إطلاقاً إنى أتكلم عن مكتبة OpenCV بأعتبارها فى جزء فرعى وسط الكتاب لأنها محتاجه كتاب لوحدها بس هحاول أديك نبذه عنها وأعرفك استخدامات المكتبة.

بداية ده الموقع الرسمى للمكتبة opencv حاول تشوفه وعارف إنى ياما جبتلك روابط لكل مكتبة وبرنامج ونستخدمه بس أنت بتكسل تفتحها 😂

المهم.. اسم المكتبة أختصار OpenCV (Open Source Computer Vision Library) والـ computer vision او الرؤية الحاسوبية ده أحد فروع علم الذكاء الإصناعى.. والرؤية الحاسوبية تهدف لجعل الكمبيوتر قادر أن يعالج الصور يتعرف على الأشخاص والـ objects اللى فيها كأنه أنسان بالضبط.

الموضوع مش غريب عليك وبتشوفه كتير فى تطبيقاتك المختلفة..زى فى الفيس بوك لما تحمل صورة بتلاقيه أتعرف أتوماتيك على الأشخاص وعملهم tag فى البوست.. ونفس الكلام فى تطبيق الكاميرا على الموبايل لما تلاقيه عمل تحديد لوجوه الأشخاص فى الصورة. دى أمثلة لتطبيقات بتستخدم تقنية الرؤية الحاسوبية computer visio..

تعتبر مكتبة openCv من المكتبات المشهوره فى المجال ده وطبعاً ميهمكش تعرف إتعملت أمتى وبلغة أيه وايه لغات البرمجة اللى بتدعمها واللى اكيد فيها البايثون بس اللى يهمك إنى أبطل رغى و نفتح الـ command line ونثبتها بالـ pip 😂



تعالا نثبتها ونكمل كلام بعدين ..

تقدر تحمل ملفات المكتبة من الموقع الرسمة releases أو تثبتها طريق pip.

```
pip install opencv-python
```

وهنحتاج مكتبة numpy اللى ثبتناها وأحنا وبنتعلم أوامر الـ pip. لو مكنتش ثبتها معانا ثبتها دلوقتى.

```
pip install numpy
```

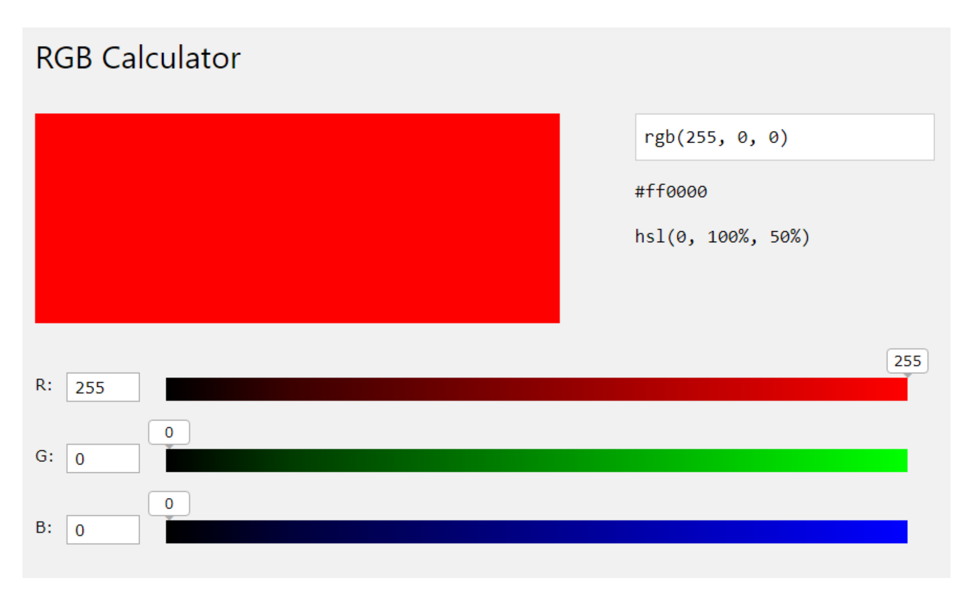
مكتبة numpy

قولنا OpenCV دى مكتبة للرؤية الحاسوبية ومعالجة الصور..يعنى هنتعامل مع صور.

طيب.. الصورة دى مجموعة من الـ pixels والبيكسل دى أصغر وحده من الصورة ولو كانت الصورة ملونة RGB كل بكسل تتكون من خليط بين 3 ألوان Red Green Blue و قيمة كل لون ممكن تخزنها فى متغير 8 بت..والـ 8 بت تقدر تشيل قيم مختلفة من 0 إلى 255 وهى دى درجة اللون اللى بتتراوح ما بين الأسود 0 والأبيض 255.

عشان تفهم الموضوع أكتر تقدر تشوف الـ RGB Calculator على موقع w3schools





مثلاً لو عاوز أعمل بيسكل لونها أحمر فى نظام RGB (R,G,B) فهى بتتكون من 3 ألوان قيمتهم كده (255,0,0) ولو عاوز بيكسل لونها أزرق (0,0,255) ولو عاوز لونها أسود كل القيم تبقى صفر (0,0,0) ولو عاوزها أبيض كل القيم تبقى (255,255,255) عشان اللون الأبيض خليط من كل الألوان زى ما انت عارف.

لعلك لحد دلوقتى ادركت إن لو الصورة ملونة فالبيكسل الوحده فيها مكونة من 3 ألوان وهحتاج 3 متغيرات أو list فيها 3 عناصر نخزن فيها قيمة كل لون.

مثلاً لو عاوز تعمل صورة لو أبيض أبعادها 4×4 يعنى أرتفاعها 4 بكسل وطولها 4 بكسل صغيرة جداً.. هيكون شكلها زى الصوره اللى على شمالك ..

```
[
  px00 px01 px02 px03
  px10 px11 px12 px13
  px20 px21 px22 px23
  px30 px31 px32 px33
]
```



- اللى قدامك دى Matrix أبعادها 4 * 4 كل عنصرفيها ليه 2 index واحد يعبر عن الصف row والتانى يعنى عن العمود col...
- مثلاً البيكسل px00 دى توجد فى الصف الاول والعمود الاول وقيمتها (255,255,255) بأعتبار أن الصوره لونها أبيض.
- لو سميت الصورة اللى قدامك دى img وعاوز اطلع البكسل px01 هيكون بالشكل ده img[0,1] يعنى طلعلى البكسل اللى فى الصف 0 والعمود 1.
- وخلى بالك فى نظام الـ RGB البيكسل دى مش مجرد رقم لكنها List مكونه من 3 عناصر (R,G,B) فى حالة صورتنا البيضه هتكون بالشكل ده (255,255,255)

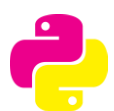
يعنى أنت عشان تخزن بكسلات الصورة عاوز مصفوفة تتكون من صفوف وأعمده بتمثل عدد البيكسلات وكل بيكسل عبارة عن list فيها 3 عناصر ودول بيمثل الألوان RGB.

ولما دورنا فى الـ data types الرئيسية فى لغة البايثون عشان نلاقى حاجة نمثل فيها الصورة وتكون بالمواصفات دى كان ممكن نستخدم list ولكن ظهرت مكتبة numpy واللى بتوفرلك multidimensional array أو مصفوفة متعددة الأبعاد وبتشغل مساحة أقل فى الرامة عن الـ objects العادية بتاعة البايثون list,tuple وبالتى استخدام مصفوفات numpy كان أفضل لتخزين الصور فى الميمورى ومعالجتها.

مكتبة numpy

هنتعلم فى مكتبة numpy اللى يفيدنا فى الموضوع ده وللتعامل مع مكتبة OpenCV.

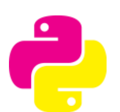
بس لو عاوز تتعمق شوف الوثيقة بتاعتها quickstart.



مصفوفات narrays

مصفوفات مكتبة numpy تسمى بـ ndarray ودى امثلة لطريقة تعريفها والتعامل معاها.

`>>> import numpy as np`	فى الأول بنستدعى مكتبة numpy
`>>> np.zeros((2,2),int)` `array([[0, 0],` `       [0, 0]])`	دالة zeros بتعمل مصفوفة كل عناصرها أصفار وبتاخد برامترين.. tuple بأبعاد المصفوفة ونوع العناصر.
`>>> np.ones((2,2),int)` `array([[1, 1],` `       [1, 1]])`	دالة ones بتعمل مصفوفة كل عناصر 1
`>>> x=np.ones((3,3),int)` `>>> x` `array([[1, 1, 1],` `       [1, 1, 1],` `       [1, 1, 1]])`	هنا عملنا مصفوفة اسمها x وفيها 3 صفوف و 3 أعمدة جميع عناصرها ونوع عناصرها int.
`>>> x.shape` `(3, 3)` `>>> x.size` `9`	shape بيرجع أبعاد المصفوفة size بيرجع عدد عناصر المصفوفة.
`>>> x[0,0]=2` `>>> x[1,1]=2` `>>> x[2,2]=2` `>>> x` `array([[2, 1, 1],` `       [1, 2, 1],` `       [1, 1, 2]])`	عشان تعدل قيمة عنصر فى مصفوفة ثنائية الأبعاد بتحدد رقم الصف ورقم العمود array[row,col] وبنعمله assignment عادى.
`>>> x[0]` `array([2, 1, 1])`	عشان تطلع صف كامل array[row_index] مثلاً x[0] بيستخرج الصف الأول.
`>>> x[:,0]` `array([2, 1, 1])`	عشان تطلع عمود كامل array[:,row_index] مثلاً x[:,0] بيرجع العمود الأول.



العمليات على مصفوفات ndarray

المصفوفات من نوع ndarray تقدر تطبق عليها العمليات الحسابية المعروفة.

<pre>>>> x=np.ones((2,2),int) >>> x=x+2 array([[3, 3], [3, 3]]) >>> y=np.ones((2,2),int)+3 array([[4, 4], [4, 4]]) >>> x+y array([[7, 7], [7, 7]])</pre>	عند جمع رقم على المصفوفة.. الرقم هيتوزع على كل عنصر فيها. تقدر تجمع مصفوفتين ليهم نفس الأبعاد والجمع هيتم بين كل عنصر مع العنصر المقابل ليه.
<pre>>>> x*y array([[12, 12], [12, 12]])</pre>	وخلي بالك إن علامة الضرب * هتضرب كل عنصر فى المصفوفة الأولي مع العنصر المقابل فى المصفوفة التانية وليس ضرب مصفوفات.
<pre>>>> y=np.ones((2,3),int)+3 array([[4, 4, 4], [4, 4, 4]]) >>> x array([[3, 3], [3, 3]]) >>> x*y ValueError:...</pre>	لو عملنا ضرب بين مصفوفة x شكلها 2*2 ومصفوفة y شكلها 3*2 هينتج خطأ لأن فى ضرب العناصر بيشترط تكون الأبعاد متساوية للطرفين.
<pre>>>> np.matmul(x,y) array([[24, 24, 24], [24, 24, 24]])</pre>	لو عاوز تعمل ضرب مصفوفات بيتم بدالة matmul ولو أبعاد المصفوفة الأولي (Row1,Col1) والمصفوفة التانية (Row2,Col2) لازم يكون Col1==Rw2.



المصفوفة ثلاثية الأبعاد

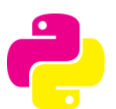
عاوزين نعمل صورة حجمها 100X100بيسكل ولونها أبيض فهتكون بالشكل ده

```
>>> import numpy as np
>>> white_img=np.ones((100,100,3),np.uint8)*255
>>> white_img[0,0]
array([255, 255, 255], dtype=uint8)
>>> white_img.shape
(100, 100, 3)
>>> white_img.size
30000
```

- عملنا مصفوفة وحايد بالدالة ones أبعادها 100 صف 100 عمود وكل عنصر فيها عبارة عن مصفوفة مكونه من 3 عناصر ones(100,100,3)
- np.uint8 بتحدد نوع المتغير unsigned int يعني رقم صحيح موجب وحجمه 8 بت.
- لما طلعنا العنصر رقم [0,0] أو البكسل الأولى رجع مصفوفة فيها 3 عناصر عناصرها [255,255,255] كل عنصر يعبر عن لون.

لحد دلوقتي فهمنا الصور بتتخزن كمتغير ازاي وان لغة البايثون بتشوفها كمصفوفة ولو عاوز تغير في لون الصورة بتغير ارقام في المصفوفة .. ولو عاوز تقص جزء من الصورة بتطلع عناصر بـ index معين.

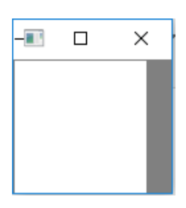
العمليات على الصور الرقمية دي اسمها Digital Image Processing أو معالجة صور ومكتبتنا الجميلة OpenCV بتوفر مجموعة من الدوال اللي بتساعدنا على معالجة الصورة والتعرف على الأشخاص والكائنات وتطبيق الفلاتر وتغير الالوان وحاجات كتير أكيد مش هتعرفها كلها هنا بس هديك نبذه مختصره بسيطه وشيقة تعرفك بالمكتبه 😁



مكتبة OpenCV

تعالو بينا نحول المصفوفات الأرقام اللى عملناها من شوية لصورة ونشوف شكلها ايه.

```
>>> import numpy as np
>>> white_img=np.ones((100,100,3),np.uint8)*255
>>> import cv2
>>> cv2.imshow('white',white_img)
```



- فى البداية عملنا المصفوفة بتاعة الصورة واسمها white_img
- بعدين أستدعينا مكتبة OpenCV واسمها cv2 هتعملها import ولو مظهرش اخطاء مشاكل يبقى المكتبة أتثبتت وتمام
- وبعدين دالة imshow دى بتعرض المصفوفة فى كـ صورة و البرامتر الاول هو اسم النافذه window اللى هتظهر فيها الصورة والتانى مصفوفة الصورة.
- بعد ما عملنا imshow طلعت الصورة الصغيره ده وده منطقى لأن حجمها 100X100 بيكسل.

انشاء صورة سوده

```
import cv2
import numpy as np
def show(img):
    cv2.imshow('Image',img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
img=np.zeros((400,400,3),np.uint8)
show(img)
```

Download script opencv.py from GitHub



- هنا عملنا مصفوفة zeros أبعادها (400,400) فنتج عنها صوره سوده.
- بعدين عملت دالة اسمها show ودى هتستخدمها فى كل الأمثلة عشان أعرض الصور بتاعتنا.
- جوة دالة show.. عندك دالة imshow دى بتعرض الصورة زى ما عرفنا.. و دالة waitkey بتخلى النافذه اللى بتظهر مستنيه أنك تضغط على أى زرار فى لوحة المفاتيح وبعدها هتنفذ الدالة destroayAllWindows اللى بتقفل النافذة
- يعنى ده أوبشن يخليك لما تدوس على أى زرار فى الكيبورد يقفل الصورة المعروضة ومن غيره الـ window هيحصلها freeze والبرنامج هيعلق

انشاء صورة خضراء

```python
import cv2
import numpy as np
def show(img):
    cv2.imshow('Image',img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

img=np.zeros((400,400,3),np.uint8)
for row in range(img.shape[0]):
    for col in range(img.shape[1]):
        img[row,col]=[0,255,0]

show(img)
```

Download script opencv1.py from GitHub



- هنا عاوزين نخلى لون الصوره أخضر green وعشان تبقى أخضر لازم شكل البكسل يبقى كده (0,255,0) عشان ترتيب الألوان فى مكتبة (B,G,R) OpenCV وخلى بالك BGR مش RGB.
- دالة zeros بتولد مصفوفة كلها أصفار (0,0,0) فهنعمل iteration نطلع بيه كل بيسكل من المصفوفة img ونغير قيمتها ونخليها (0,255,0)
- وخلى المتغير shape بيرجع أبعاد المصفوفة, shape[0] ده عدد الصفوف و shape[1] ده عدد الأعمدة..فبعمل iteration عليهم.

حفظ الصوره دالة imwrite

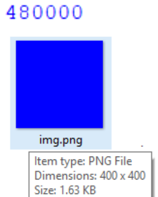
فى المثال ده هنعمل صورة زرقاء كل بيكسل فيها قيمته (255,0,0) ونحفظها على الجهاز.

<pre>import cv2 import numpy as np img=np.zeros((400,400,3),np.uint8) for row in range(img.shape[0]): for col in range(img.shape[1]): img[row,col]=[255,0,0] cv2.imwrite('img.png',img)</pre>	قولنا ترتيب الالوان فى البيكسل (B,G,R) وعشان نخلى الصورة زرقاء هنخلى البيكسلات فى المصفوفة img بالشكل ده (255,0,0) هنحفظ الصورة img باسم img.png وبدالة imwrite اللى بتاخد برامترين الاول اسم الصوره والأمتداد والتانى المصفوفة اللى بتمثل الصورة.

Download script opencv2.py from GitHub

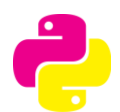


```
>>> img.size
480000
```



الصـورة اللـى أحنا عـملـناها أبعادها 400 X 400 يعنـى عـدد عناصـرها size 3×400×400 وكل عنصـر فيـها يمثل 8 بت يعنـى حجـم الصـورة لما تخزنها المـفروض يكـون 480,000 بايت = 496 كيلـو.

- بس الحقيـقة إن حجـم الصـورة طلع أقل من كيلـو 2..تخيل صـورة لونها أزرق حجـمها 400 بيـكسل تتخـزن فى نص ميـجا طيـب لو فريم من فيـديو 4k أبعاده 3840 × 2160 فحجـمه هيكـون كام وحجـم الفيـديو اللـى فيـه أكـتر من 20 فريم فى الثانيـه لو مـدته ساعـه ؟!.
- عشـان كـده بيـتم عمل encoding للصـور أو الفيـديو بالصـيغ المختلفـة اللـى بتشـوفها زى png,jpg واللـى بتعمل ضـغط compress للصـورة بطـريقـة معـينة تقلـل من عـدد الـ bits اللـى هتتخـزن علـى الجهاز بـدل ما تخزن عـدد كـبير متـكـرر من نفـس الـ pixels.



قراءة وعرض صورة imread

هتحفظ الصورة دى جنب الـ script اللى شغال عليه logo.png وهتقراها بمكتبة .OpenCV

```
import cv2
import numpy as np

def show(img):
    cv2.imshow('Image',img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

main = cv2.imread('logo.png')
scaled = cv2.resize(main, (0,0), fx=4, fy=4)
gray = cv2.cvtColor(scaled,cv2.COLOR_BGR2GRAY)

show(scaled)
show(gray)
```

Download script opencv3.py from GitHub

- هنا دالة imread بتاخد مسار الصورة اللى عايز تقراها وبترجعها فى ndarray.
- دالة resize بتغير حجم الصورة واستخدمناها عشان نعمل scale أو تكبير للصورة الأصلية main بنسبة 4 مرات للطول والعرض بحيث خلينا البرامترات fx=4,fy=4 .
- دالة cvtColor أستخدمناها عشان نحول الصورة الملونة من BGR إلى صورة أبيض وأسود GRAY .



صور Gray Scale

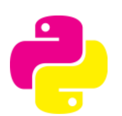
قبل كده أتكلمنا عن الصور الملونة بنظام RGB أو BGR وهو إن كل بيكسل فى الصورة بتتكون من 3 ألوان Blue,Green,Red ولما تتخزن الصورة فى ndarry كل بيسكل بيعبر عنها مصفوفة فيها 3 قيم.

- فيه نوع تانى من الصور وهو الأبيض والأسود Grayscale وهو عبارة عن تدرج بين درجة اللونين الأسود 0 والأبيض 255 وبالتالى في النظام ده البيكسل يعبر عنها قيمة وحده بترراوح بين 0:255 وليس مصفوفة فيها 3 قيم زى نظام BGR.

```
>>> gray.shape
(396, 328)
>>> gray.size
129888
>>> gray[0,0]
0
>>> gray[100,100]
89
```

- شغل السكريت اللى فات وقبل ما تقفل الـ Interpreter جرب تشوف الـ attributes بتاعة المصفوفة gray أو الصور الأبيض وأسود.
- Shape هترجع (width ,height) ومفيش البعد الثالث.
- دالة size عبارة عن حاصل ضرب الطول*العرض = 328*396 وبالتالى توقع إن حجم الصورة grayscale أصغر من RGB اللى فى حالته بتضرب فى 3.
- كل عنصر فى الدالة عبارة عن قيمة وحده 8 بت وليس مصفوفة فيها 3 عناصر.

ممكن تسأل إيه فايدة نظام الـ Grayscale وليه مبتعاملش مع الصور الملونة ديماً 🤔!؟.
أعتقد إنك شوفت إن فى حال Grayscale حجم المصفوفة بيقل 3 أضعاف وده بيقلل من المساحة اللى هتشغلها فى الرامه و بيزود من سرعة المعالجة لو عاوز تنفذ ألجوريثمات وعمليات معينة على المصفوفة.



الرسم Drawing

مكتبة OpenCV فيها مجموعة دول لرسم العديد من الأشكال زى الخطوط والدوائر والمستطيلات والأشكال البيضاوية.. بالإضافة لأمكانية كتابة نصوص على الصور.

كل دول الرسم زى line,rectangle,circle بتحدد ليها الصورة اللى عاوز ترسم عليها والإحداثيات اللى هترسم عندها بالإضافة للون والسمك بتاع كل شكل.

رسم خط line

```
cv2.line(image,(x1,y2),(x2,y2),color,thickness)
```

رسم مستطيل rectangle

```
cv2.rectangle(image,(x1,y2),(x2,y2),color,thickness)
```

رسم دائرة circle

```
cv2.circle(image,(x,y),radius,color,thickness)
```

إضافة نص putText

```
cv2.putText(image,Text,(x,y),Font,Size,color,thickness)
```



وده سكريت بيرسم كل الأشكال اللى أتكلمنا عنها.

```
import cv2
import numpy as np

def show(img):
    cv2.imshow('Image',img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

img=np.ones((500,500,3),np.uint8)*255

red   = (0,0,255)
green = (0,255,0)
blue  = (255,0,0)

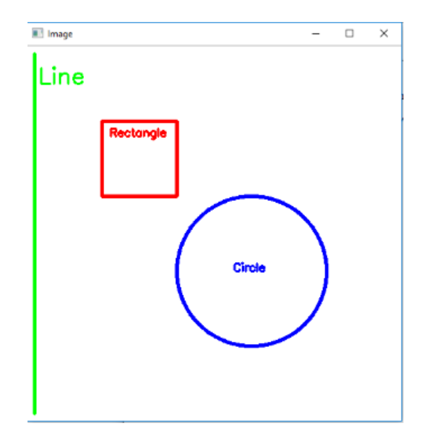
cv2.line(img,(10,10),(10,490),green,4)
cv2.putText(img,'Line',(15,50),cv2.FONT_HERSHEY_SIMPLEX,1,green,2)

cv2.rectangle(img,(100,100),(200,200),red,4)
cv2.putText(img,'Rectangle',(110,120),cv2.FONT_HERSHEY_SIMPLEX,0.5,red,2)

cv2.circle(img,(300,300),100,blue,4)
cv2.putText(img,'Circle',(275,300),cv2.FONT_HERSHEY_SIMPLEX,0.5,blue,2)

show(img)
```

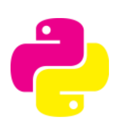
Download script opencv4.py from GitHub



ده شكل الصورة الناتجة من السكريت .. وعشان

تعرف أكتر عن الرسم بالـ OpenCv كمل من هنا

tutorial_py_drawing_functions



التعرف على الوجوه Face Detection

من الحاجات الجميلة اللى بتشوفها في تطبيقات الكاميرا أو حتى على الفيس بوك هى خاصية التعرف على وجوه الأشخاص face detection المقصود هنا إن عن طريق معالجة الصورة هقدر أعرف إذا كان الصورة دى فيها وجه face ولا لأ أو فيها كام شخص.

بالنسبة للتعرف على الوجه ده تبع مين برده من الحاجات اللى تقدر تعملها بالـ opencv بس للأسف هسيبك تعملها لوحدك لو مهتم يعنى لأنى عاوز أخلص الكتاب 😂

طريقة Haar-cascade

من طرق التعرف على الوجوه أو الـ objects فى الصور هى طريقة haar cascade..

فى البدايه لو معانا صورة digital لما أتعامل معاها بتبقى عبارة عن مصفوفة من الأرقام.

أنا عاوز أعرف هل تجمع من الأرقام دى فى مساحة من الـ pixles او window معينة يمثل صورة وجه Face مثلاً ؟!

الحل إنى أحاول اطلع الخصائص features بتاعة وجه الأنسان اللى بتظهر فى الصورة.. فمثلاً المنطقة بتاعة العنين والحواجب لونها بيبقى غامق اكتر من منطقة الخدود.. يعنى مجموع الأرقام بتاعة البيسكلات فى منطقة pixel intensity هتختلف عن منطقة أخرى حسب تدرجها من الأبيض للأسود لو الصورة grayscale مثلاً فبقدر أتوقع الجزء ده بيمثل إيه منطقة أيه فى الوجه.



والطريقة اللى بقدر أصنف بيها أجزاء الصورة اسمها haar-like feature والاسم مأخوذ من عالم الرياضيات Alfréd Haar.

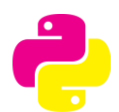
بالطريقة دى اقدر أطلع الخصائص features بتاعة العنصر اللى عاوز أعملها detection وجه الانسان مثلاً .. خلاص عندى خصائص ممكن أقسم الصورة بتاعى اجزاء windows وأطبق عليها الخصائص دى وأشوف فيها Face ولا لأ وبكده عن طريق برنامج وحسابات رياضية لأرقام وبيكسلات أقدر أخلى الكمبيوتر يتعرف على وجود أشخاص فى الصورة من عدمه وهو ده الذكاء الإصطناعى يا جماعة😁.

المهم اللى عملوا الألجوريثم لقيوا إن عندهم أكتر من 6000 feature مختلفة المفروض يطبقوها على أجزاء الصورة عشان يقولك الجزء ده فيه Face ولا مفيهوش وطبعاً العملية دى هتاخد وقت طويل.

فقرروا يقسموا العملية دى مراحل متتالية cascade stages.. فى كل مرحلة هيدور على مجموعة من الخصائص ويشوفها هل موجودة ولا لأ.. لو مش موجودة فى الجزء ده خلاص مش هيكمل مطابقة للخصائص وعرف إن دى مش صورة وجه وبالتالى أختصر الوقت بشكل كبير.

تمام.. كل الكلام ده عشان عاوز أفهمك معنى haar cascade اللى هنستخدمها بدل ما نطبق الأكواد على طول وأقولك هنستخدم طريقة مش عارف ونشوف النتيجة بس.

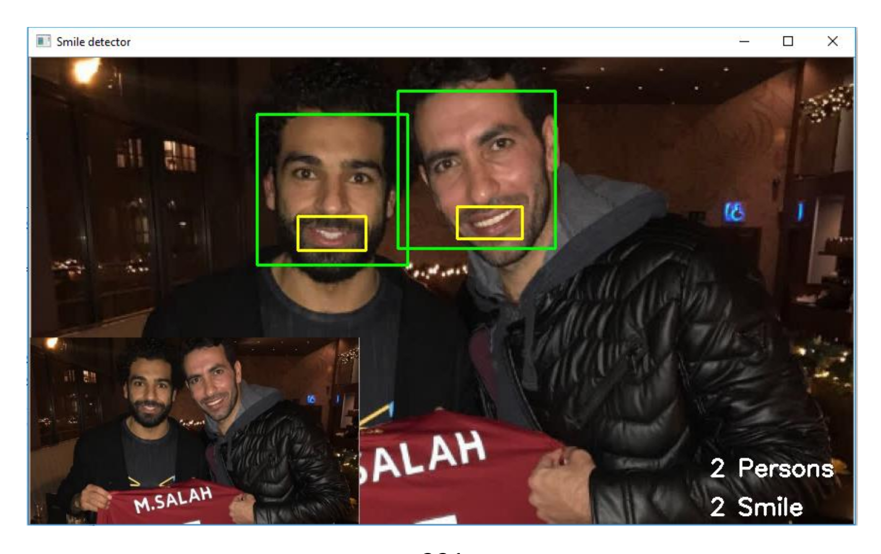
طبعاً لو عاوز تفهم الموضوع بالتفصيل تقدر تشوف الـ tutorial دى على الموقع الرسمى tutorial_py_face_detection.



Haar-cascade Detection in OpenCV

- يبقى عشان تخلى مكتبة OpenCV تتعرف على أى object مهما كان بتحتاج للـ classifier بيبقى فيه الخصائص features للـ object ده عشان يطبقها على الصورة اويحاول يتعرف عليه.
- مكتبة openCv بيجى معاها classifiers جاهزة لو كنت مثبت المكتبة عن طريق pip هتلاقيهم فى الفولدر site-packages/cv2/data .
- الـ classifires دول بيخلوك تقدر تتعرف على أجزاء كتير فى الصورة زى العنين والوجه وجسم الانسان ورخصة العربية وحاجات تانيه ومكن تحملهم من هنا haarcascades

و دلوقتى عاوزين نعمل سكريت يتعرف الوجوه والأبتسامات فى الصورة وهحمسك أكتر عشان الكود طويل وممكن تخاف منه ده اللى بيعمله السكريت 😂.





- فى المكان اللى هتحفظ فيه السكريت هتحتاج تضيف ملفين من haarcascade الأول فيه خصائص الوجه haarcascade_frontalface_default.xml والتانى للإبتسامة haarcascade_smile.xml ده بالإضافة للصور اللى فيها أشخاص ودول اللى أستخدمتهم img1.jpg و img2.jpg طبعاً تقدر تختار أى صور من عندك.



```
import numpy as np
import cv2

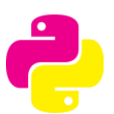
def show(img):
    cv2.imshow('Image',img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

img  = cv2.imread('img1.jpg')
small=cv2.resize(img.copy(),(0,0),fx=0.4,fy=0.4)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_cascade   = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smile_cascade  = cv2.CascadeClassifier('haarcascade_smile.xml')
faces = face_cascade.detectMultiScale(gray)
s=0
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
    face_gray  = gray[y:y+h, x:x+w]
    face_color = img[y:y+h, x:x+w]
    smiles = smile_cascade.detectMultiScale(face_gray)
    for (ex,ey,ew,eh) in smiles:
        cv2.rectangle(face_color,(ex,ey),(ex+ew,ey+eh),(0,255,255),2)
        s=s+1
        break
height=img.shape[0]
width=img.shape[1]
x_offset=0
y_offset=gray.shape[0]-small.shape[0]
img[y_offset:y_offset+small.shape[0], x_offset:x_offset+small.shape[1]] =
small
cv2.putText(img,'%s Persons'%len(faces),( width-150,height-50),
cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
cv2.putText(img,'%s Smile'%s,(width-150,height-10),cv2.FONT_HERSHEY_SIMPLEX,
0.8,(255,255,255),2)
show(img)
```

Download script opencv5.py from GitHub



- فى الأول قرينا الصورة اللى عاوزين نتعرف على الوجوه والابتسامات فيها واللى كان اسمها هنا img1.jpg
- بعدين أخدنا نسخة من الصورة وعملنا لها resize وسميناها small ودى الصورة الأصلية اللى هتظهر فوق الصورة اللى هحدد عليها الوجوه.
- و حولت الصورة img لـ gray
- بعدين هنقرى الـ classifires بتوعنا face_cascade وsmile_cascade.
- دالة detectMultiScale عطيناها برامتر واحد وهو الصورة gray عشان ترجع list اسمها faces فيها عدد lists يمثل عدد الوجوه اللى تعرف عليها وفى كل list فيه الأحداثيات x,y والطول والعرض أقدر منهم أرسم rectangle وأحدد الوجه اللى أتعرف عليه.
- عشان أدور على الـ smile مش محتاج ادور فى الصورة الكبيرة ..فبوفر وقت وأدور فى جوة الـ face اللى طلعته مسبقاً فبيعمل crop للوجه الى جمعه فى الصورة الـ gray وسماه face_gray وفى الصورة الملونة سماه face_color.
- طبعاً الوجه الـ gray بدور جواه على smile بدالة detectMultiScale التابعة للـ smile_cascade والـ face_color دى برسم عليه المستطيل الأخضر.
- وفى النهاية بحط نصين puttext على الصورة الأصلية اللى اسمها img بعدد الأشخاص وهو len بتاع الليسته faces وعدد الأبتسامات وده حسبناه فى المتغير s.



إلتقاط فيديو من الكاميرا Capture Video from Camera .

ممكن تقولى أنا عاوز أطبق السكريبت اللى فات على كاميرا بتصور لايف بحيث إن الكاميرا لما تتعرف على و أشخاص فى المكان مثلاً نخلى البرنامج يبعت تنبيه أو يبعت صورة الأشخاص دول فى إيميل وكده يعني...مفيش مشاكل ممكن نعملها وده سكريبت بيوضح أزاى مكتبة OpenCv بتاخد صور من الكاميرا وبتعرض

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
while(True):
    ret,img = cap.read()
    cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

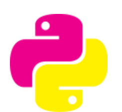
Download script opencv6.py from GitHub

- فى البداية دالة VideoCapture بتاخد برامتر واحد وهو الـ video source أو مصدر الفيديو اللى بتتعامل معاه.. أما كاميرا وبتكتب رقمها فى الغالب الكاميرا الأفتراضية بتاعة اللاب رقمها 0 وممكن يكون ملف فيديو فبتكتب اسمه.
- بتفتح جملة تكرار while عشان تقرى الصور من الكاميرا.
- الكاميرا بتسجل الفيديو على شكل frames وهو صورة كل جزء من الثانيه حسب نوع الكاميرا وجودة الفيديو طبعاً .



- دالة read بترجع متغيرين الاول متغير bool بيكون True لو الـ frame أتصور بشكل صحيح.. والمتغير التانى الـ frame أو الصورة وهنا سميتها img.
- بعمل عرض للفريم بدالة imshow كالمعتاد.
- جملة الشرط if cv2.waitKey(1) & 0xFF == ord('q') بتخرج من جملة التكرار وبتقفل تصوير الفيديو لو المستخدم داس على حرف من لوحة المفاتيح وهنا حددنا حرف q.
- تقدر تجرب السكريت وتشغله على الكاميرا وتشوفه بيصور فيديو ازاى.

وللمزيد تقدر تشوف الـ tutorial دى py_video_display



التعرف على الوجوه فى الفيديو

هندمج السكريتين الى فاتوا بحيث إننا نخلى البرنامج يتعرف على الوجوه والأبتسامات

من الكاميرا مش من صورة ثابتة زى ما عملنا.

```
import numpy as np
import cv2

face_cascade   = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smile_cascade  = cv2.CascadeClassifier('haarcascade_smile.xml')
cap = cv2.VideoCapture(0)

def detect_persons(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray)
    s=0
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),1)
        face_gray  = gray[y:y+h, x:x+w]
        face_color = img[y:y+h, x:x+w]
        smiles = smile_cascade.detectMultiScale(face_gray)

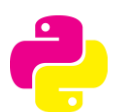
        for (ex,ey,ew,eh) in smiles:
            cv2.rectangle(face_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)
            s=s+1
            break

    cv2.putText(img,'%s
Persons'%len(faces),(20,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1)
    cv2.putText(img,'%s
Smile'%s,(20,50),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1)
    return img
while(True):
    ret,img = cap.read()
    img=detect_persons(img)
    cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows
```

Download script opencv7.py from GitHub

تقدر تجرب السكريت ده على كاميرة اللاب أو أى كاميرا usb متوصلة بالكمبيوتر

بس هتحتاج تغير رقم الكاميرا فى الكلاس VideoCapture.



استخلاص النصوص من الصور OCR

التعرف على الحروف الضوئية Optical character recognition هو عملية تحويل الكتابة الموجودة فى الصور إلى نص وهو من المجالات الشيقة التابعة للرؤية الحاسوبية computer vision والذكاء الإصطناعي Artificial Intelligence .

وتقدر تعرف المزيد عن الموضوع ده من الويكيبيديا Optical character recognition

الموضوع هيكون مهم معاك لو عندك كتاب مصور او pdf وعاوز تحوله لصيغة نصية أو word.. فإما تلجأ للطريقة الكتابة اليدوية وتجيب حد يكتبهولك أو تستعين بأى برنامج يحولك الكتابة في الصور لنصوص قابلة للتعديل في لحظات بدقة تعتمد على البرنامج أو الـ OCR Engine المتسخدم وبرده بيعتمد على جودة الصورة ومدى وضوحها.

لكنه وعلى كل حال بيوفر وقت وجهد بشكل كبير.

برنامج tesseract ocr

ده عبارة عن برنامج مفتوح المصدر للتعرف على النصوص text recognizer بيدعم أكتر من 100 لغة من ضمنهم اللغة العربية و كبرنامج command line أو مكتبات تقدر تضمنهم فى برامجك بلغة ++C او تطبيقات android و ios.

طريقة تحميل وتثبيت tesseract حسب نظام تشغيلك من هنا tesseract-ocr

ولو كنت من مستخدمين فتقدر تحمل نسخة قابلة للتثبيت من هنا tesseract.



التعامل مع برنامج tesseract

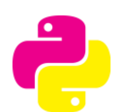
- أياً كان نظام تشغيلك بعد ما تثبت البرنامج أفتح الـ command line وأكتب فيه tesseract عشان تتأكد إن البرنامج إتثبت..

```
C:\Users\Mahmoud>tesseract
Usage:
  tesseract --help | --help-psm | --help-oem | --version
  tesseract --list-langs [--tessdata-dir PATH]
  tesseract --print-parameters [options...] [configfile...]
  tesseract imagename|stdin outputbase|stdout [options...] [configfile...]
```

- لو كنت من مستخدمين ويندوز ونزلت البرنامج وثبته هتلاقى الملفات بتوعه فى المجلد Tesseract-OCR جوه Program files أو Program Files (x86) حسب نسخة البرنامج اللى ثبتها.
- بس فى البداية هتفتح الـ CMD وتكتب فيه tesseract ولو متعرفش على الأمر يبقى هحتاج تحط مسار المجلد بتاع البرنامج C:\Program Files (x86)\Tesseract-OCR فى الـ path بتاع الـ environment variables زى ما عملنا فى برنامج ffmpeg.

بأعتبار إنك ضفته وتمام.. هترجع لمجلد Tesseract-OCR هتلاقى جواه مجلد tessdata ده المجلد اللى فيه training data للغات اللى بيقدر يتعرف عليها البرنامج.

لو ملقيتش اللغة اللى عاوزها هتنزلها من هنا Data-Files وتنسخها فى مجلد tessdata.



تحويل صورة إلى نص من خلال command line

طريقة استخدام البرنامج..

```
tesseract image_name output_file
```

- وبالأمر ده البرنامج هيحاول يستخرج النص من الصورة اللى بتحدد اسمها ويخزنه فى ملف output_file باللغة الأفتراضية اللى هى الإنجليزية eng.

لتحديد اللغة

```
tesseract image_name output_file -l language
```

ودي قائمة باللغات اللى بيدعمها البرنامج Data-Files

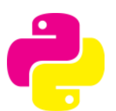
هناخد مثال على الـ command line .. هنجيب الصورة اللى عاوزين نحولها لنص و فى المثال ده أنا عملت سكرين شوت لجزء من الوثيقة بتاعة المكتبة وحفظتها باسم ocr.png.

Brief history

Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some C++izing in 1998. In 2005 Tesseract was open sourced by HP. Since 2006 it is developed by Google.

فلو هنجرب على الصورة ocr.png

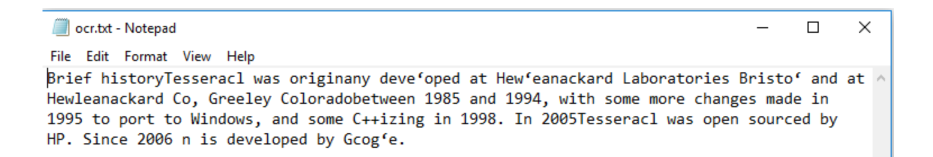
```
>tesseract ocr.png ocr
```



- ocr.png دى الصورة بتاعتنا و ocr ده الملف النصى اللى هيخزن فيه النص اللى هيطلعه من الصوره .. وبتكتب اسم الملف فقط مش بتكتب ocr.txt.
- مقارنة بين شكل الصورة والنص اللى أستخرناه منها واللى فيه نسبة خطأ.

Brief history

Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some C++izing in 1998. In 2005 Tesseract was open sourced by HP. Since 2006 it is developed by Google.



ocr.txt - Notepad

File Edit Format View Help

Brief historyTesseracl was originany deve‘oped at Hew‘eanackard Laboratories Bristo‘ and at Hewleanackard Co, Greeley Coloradobetween 1985 and 1994, with some more changes made in 1995 to port to Windows, and some C++izing in 1998. In 2005Tesseracl was open sourced by HP. Since 2006 n is developed by Gcog‘e.

استخدام tesseract مع لغة البايثون

برنامج tesseractocr له أكتر من wrapper بأغلب لغات البرمجة المشهورة ومنها البايثون اللى ليها مكتبة مشهورة اسمها pytesseract.

- تثبيت المكتبة باستخدام pip

```
pip install pytesseract
```

- بتحتاج تثبت معاها مكتبة تانيه اسمها pillow ودى بتقرى الصورة اللى عاوز تستخرج منها النص.

```
pip install pillow
```



بعد ما هتثبت pytesseract و pillow وياريت تكون متأكد من تثبيتك لبرنامج tesseract ocr في الـ command line تقدر تشغل السكريت ده

```
from PIL import Image
from pytesseract import image_to_string

img  = Image.open('ocr.png')
text = image_to_string(img)

print(text)
```

Download script ocr.py from GitHub

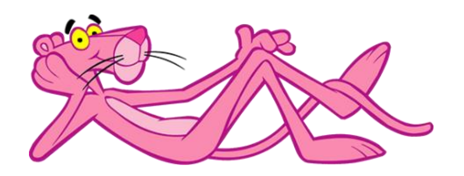
- هنا عاوزين نستخرج النص من نفس الصورة ocr.png
- دالة open بتقرى الصورة في object اللي بنمرره للدالة image_to_string عشان تستخلص النص منه.

عشان تعرف اكتر عن المكتبة من هنا pytesseract.

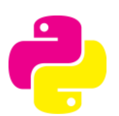
وعشان تعرف أزاي تستخدم البرنامج مع الصورة اللى فيها كتابة بالعربي أو تعمل training للبرنامج بحيث إنك تحسن أداءه فلازم تدرس الموضوع كويس من الـ documentation بتاعة البرنامج tesseract



الفصل 6 – تطبيقات الويب Python Web Application



الفهرس

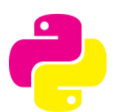


فى الفصل ده هنتعلم بعض طرق استخدام لغة البايثون فى تطبيقات الويب وازاى نستفيد بالحاجات اللى أتعلمناها أننا نعمل موقع بلغة البايثون ونستضيفة على سيرفر ونقدر نوصله من أى مكان.

أزاى بتشتغل المواقع

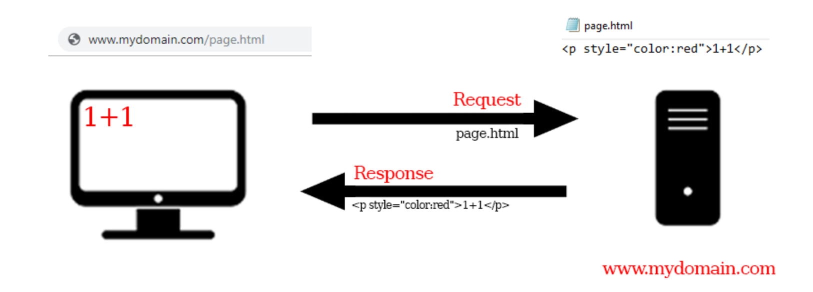
بس فى الاول عاوزين نعرف مواقع الأنترنت شغاله أزاى 🤔؟

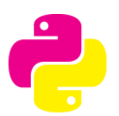
- الموقع بتاعك عشان يبقى متاح لأى حد على الأنترنت بيتحط على جهاز كمبيوتر متوصل بالأنترنت اسمه السيرفر server. السيرفر ده ليه عنوان ثابت ip address بيبقى مربوط بعنوان الموقع الـ domain name.
- لما تفتح المتصفح وتكتب عنوان الموقع domain name المتصفح بيعمل حاجة اسمها http request يعنى بيبعت طلب للسيرفر ده عن طريق بروتوكول اسمه http يقوله هاتلى المحتوى بتاع الموقع أو الصفحة دى.
- السيرفر لو لقى الصفحة هيرد عليه بالـ response وهو المحتوى بتاع الصفحة على شكل نص plain text..النص ده بيوصل للمتصفح اللى بيقوم بتحويله إلى الشكل المرئى Graphical اللى فيه صور ورسومات وألوان وزراير والحاجات اللى بتتعامل معاها فى صفحات الأنترنت.
- محتوى الموقع ده أما يكون محتوي ثابت static وهو صفحة مكتوبة بلغة html ودى لما المتصفح يبعت request السيرفر بيبعتله الكود بتاع الصفحة زى ما هو زى ما أنت خزنته على السيرفر.

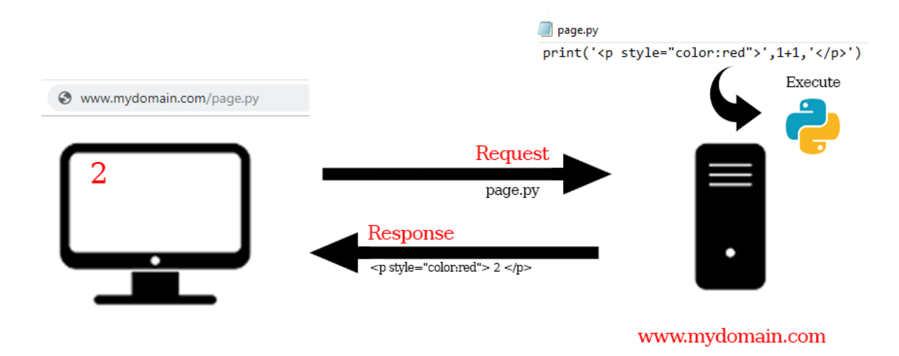


- طيب لو عاوز تعمل موقع يخزن داتا فى قاعدة البيانات ويتعامل مع نظام الملفات ويعمل تسجيل دخول وتسجيل أعضاء ومواضيع وغيره. فى الحالة دى الموقع بتاعك مش مجرد صفحة html ميته لما تطلبها السيرفر بيبعتهالك زى ما هى .. الموقع هيتعامل مع السيرفر وقواعد البيانات اللى عليه, وعشان نعمل مواقع زى دى بنحتاج مع الـ html و لغات التصميم لغات برمجة اسمها server side.
- فى الحالة دى لما تعمل request للسيرفر باسم الموقع بتاعك.. السيرفر مش هيبعت المحتوى بتاع الموقع على طول لأن المحتوى ممكن يكون كود بلغة البايثون واحنا مش عايزين الكود عايزين النتيجة بتاعته.
- فالسيرفر لما يجيله الـ request هياخد الكود بتاع الموقع اللى هو عبارة عن سكريبت بلغة معينة زى php,Python,Ruby,Ring و يعمله execute فى الـ interpreter الخاص باللغة ويبعت النتيجة فى الـ response ودى اللى هتشوفها انت فى المتصفح.

الصورتين دول بيوضحوا الفرق لما تفتح صفحة موقع html ولما تفتح موقع معمول بلغة البايثون.







- لاحظ الفرق بين الصورة دى والصورة اللى فوقيها .. لما السيرفر جاله request باسم الصفحة اللى page أمتدادها html. بعت الكود اللى فيها زى ما هو.
- ولما بعتناله request باسم الصفحة اللى عنوانها py. أحنا معرفينه أنه لو جه request بألامتداد ده ياخد الصفحة ويعملها execute فى المفسر بتاع لغة البايثون ويبعت النتيجة.

طيب يعنى انا لو عاوز أعمل تطبيق بسيط جداً مثلاً يجمع رقمين هل أعمله بالبايثون🤔؟

- مش لازم .. لو تعرف لغة الـ javascript والاستخدام الشائع ليها إنها لغة client side بتتفسر فى المتصفح.. فتقدر من خلالها كل العمليات الحسابية والمنطقية وأى حاجة ليها علاقة بمكونات صفحة html أو المتصفح زى DOM و BOM وكل حاجة تتخيلها على المتصفح فمفيش داعى إنك تستخدم لغة البايثون.



طيب أمتى تستخدم لغة البايثون أو أى لغة server side 🤔

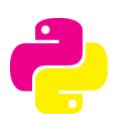
- الإجابة هتستخدمها فى حالة إنك عاوز تعمل موقع يتعامل مع database او file system او sessions أو cookies والحاجات اللى متقدرش تعملها بالجافا سكريت كلغة client side.

الاستضافات Hosts

الاستضافة هى المساحة اللى بتشتريها على السيرفر المتصل بالأنترنت عشان تحط الملفات بتاعتك وبتبقى مربوطة بالـ domain name بحيث إن أى حد يقدر يوصل للموقع من خلاله ويبقى عليها البرنامج اللى (برده أسمه سيرفر) اللى لما يجيله request بعنوان الصفحة ينفذها سواء كانت صفحة php او python او asp.net .

و الاستضافات اللى تقدر تنشر عليها تطبيقات بلغة البايثون نوعين :

- اما تبقى cloud hosting او vps وأمثلة للإستضافات دى زى Heroku و OpenShift وAzure أو PythonAnyWhere وغيرهم كتير.
- والنوع التانى هو إستضافات shared hosting ودى فى الغالب بيبقى متثبت عليها سيرفر apache واللى بيدعم أكتر من interface و موديول يسمحوا له بتفسير لغة البايثون.. وامثلة للأستضافات الـ shared hosting اللى بتدعم لغة البايثون زى GoDaddy و Bluehost وغيرهم كتير.



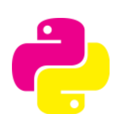
ويب سيرفر web servers

الويب سيرفر Web server مكون من جزئين ..

- جزء هاردوير وهو جهاز كمبيوتر المتوصل بالأنترنت وشغال 24 ساعة وبتحجز عليه مساحة تخزن عليها ملفات الموقع بتاعك واسمها الأستضافة host وبيديك عنوان رقمى ip address بتربطه باسم الموقع domain name عن طريق خدمة DNS.
- تانى حاجة جزء السوفت وير وهو البرنامج اللى بيشتغل على السيرفر عشان ينظم عمليات الإتصال بين السيرفر server والمستخدمين clients عن طريق بروتوكولات http/https وهو ده اللى بيحدد طريقة التواصل بين المتصفح بتاعك والكمبيوتر اللى عليه الموقع و هو اللى بيستقبل الـ requests وبيرد بالـ response اللى بتشوفها فى المتصفح.بالأضافه إنه بيتعامل مع ملفات الموقع حسب نوعها لو صفحات html بيبعتلك المتحوى بتاعها زى ما هو ولو ملفات بأى لغة برمجة php,Python,Ruby بيعملها execute وبعدين يبعتلك النتيجة فى الـ response..

من أمثلة سيرفرات http المشهورة سيرفر أباتشى apache وسيرفر nginx .

أنت كمبرمج عشان تعمل سكريتات بلغة البايثون وتشتغل فى المتصفح فى مرحلة التطوير او التجريب بتحتاج برنامج سيرفر محلى هتثبته على جهازك والمرحلة التانية إنك تعمل نشر للموقع deploying و تستضيف السكريبتات على الـ hosts اللى بتدعم لغة البايثون زى ما قولنا من شوية.



تقنية CGI

من التقنيات القديمة المستخدمة لتشغيل تطبيقات لغة البايثون و لغات البرمجة التانيه على الويب هى Common Gate Interface CGI .. وزى ما ذكرت مسبقاً لما تبعت request للسيرفر هيرد عليك فى الـ response بالمحتوى بتاع صفحة الويب لو صفحة html.. وإن كانت سكريت للغة برمجة هيبعت السكريت ده للـ interpreter ويرجعلك النتيجة بتاعته يعنى السيرفر مينفعش يبعتلك السكريت بتاعك بصورته النصية.. وإن السيرفر بيعمل كوسيط interface بين المستخدم والـ interpreter اللى بيفسر السكريت ومن هنا جاءت التسمية.

سيرفر أباتشى Apache server

عشان نقدر نطور مواقع بلغة البايثون ونجربها على المتصفح بتاعنا أو نخليها متاحة لأى مستخدم على الانترنت بنحتاج برنامج http server ومن أشهر البرامج دى سيرفر أباتشى apache.

عشان تحمل أباتشى على ويندوز كالعاده فى مواقع البرامج مفتوحة المصدر بتلاقى السورس كود فى الموقع الرسمى وهتحتاج تعمله build بنفسك.. فأحنا مش هنعمل كده ونشوف أى حد عامل releases منه لنظام التشغيل بتاعنا.. و هتلاقى أكتر من برنامج ويب سيرفر معتمدين على سيرفر الأباشتى زى wamp server وxampserver أو appserve أو Bitname .



طبعاً مستخدمين php عارفين البرامج دى وبيستخدموها أو واحد منها كسيرفر فى برمجة المواقع بالـ php لأنها عبارة عن برنامج apache server ومتضاف عليها موديولات او interfaces للغة php.

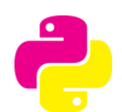
أنا هستخدم برنامج اسمه apachehaus وهو نسخة خفيفة لسيرفر الأباتشى.

البرنامج ده مش الأفضل ممكن تستخدم أى برنامج تانى من اللى ذكرتهم فوق مفيش مشكلة.. المهم تشغل سيرفر أباتشى وتعمل الأعدادات اللى هنعملها بعد شوية.

- لو حملت apachehaus.. هتفك ضغط الملف هتلاقى مجلد اسمه Apache24 هتنسخة مثلاً فى البارتيشين C وهيكون جواه الفولدرات دى ..

bin	cgi-bin	conf
error	htdocs	icons
include	lib	logs
modules	ABOUT_APACHE.txt	CHANGES.txt
INSTALL.txt	LICENSE.txt	NOTICE.txt
OPENSSL-NEWS.txt	OPENSSL-README.txt	README.txt

- مجلد bin وده اللى فيه الـ binaries أو الملفات التنفيذية اللى هنشغل منها السيرفر.
- مجلد cgi-bin ده هنحط فيها سكريبتات لغة البايثون اللى عاوزين نشغلها فى المتصفح عن طريق تقنية CGI.
- مجلد conf فيه ملف httpd.conf بنزبط منه إعدادات السيرفر وهنعمل فيه تعديل.



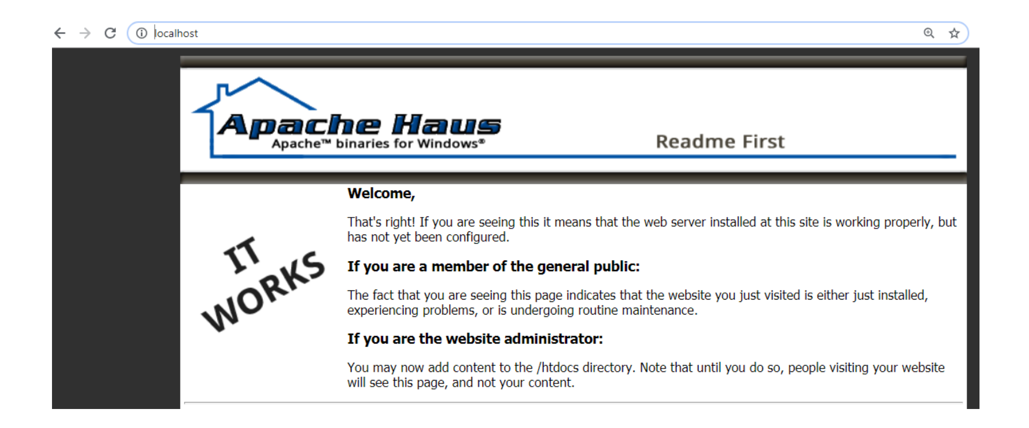
تشغيل سيرفر Apache

- عشان تشغل البرنامج هتدخل بالـ command line في الفولدر bin أياً كان مكانه.
- وهتكتب اسم الملف اللي بيشغله httpd هتسيب الـ command line مفتوح ومتقفلهوش لأنك لو قفلته السيرفر هيوقف.

```
>cd apache24/bin

Apache24\bin>httpd
```

- وأخيراً هتفتح المتصفح وتكتب فيه localhost وهتظهر معاك الصفحة دى.



- فى برامج سيرفرات تانية زى wamp و xamp و Bitname وليها واجهة رسومية وطريقة تشغيل أسهل من كده لو مش حابب البرنامج ده.



إعداد سيرفر أباتشي Apache configuration

- عشان تقدر تخلي سيرفر أباتشي يشغل سكريتات لغة البايثون في المتصفح لازم تعمل تعديل بسيط في ملف httpd.conf الموجود في الفولدر conf .
- أفتح الملف ده ودور على السطر ده AddHandler cgi-script

```
# To use CGI scripts outside of ScriptAliased directories:
# (You will also need to add "ExecCGI" to the "Options" directive.)
AddHandler cgi-script .cgi .pl .py
```

- هتشيل الكومنت # من السطر AddHandler وتضيف الأمتداد .py.
- اللي عملناه هيخلي السيرفر لما يشوف أي ملف أمتداده .cgi او .py وده سكريت بايثون أو . pl وده سكريت للغة perl مالناش دعوه بيه..ساعتها مش هيبعتلك المحتوى بتاعه.. هيبعته للـ interpreter بتاعه يعمله execute زي ما هتحدد في أول السكريت وهنشوف أزاي.

بعد ما تعمل الخطوات دي هتقفل السيرفر وتشغله تاني بأنك هتقفل الـ command line اللي شغلنا فيه السيرفر من شوية وتدوس Ctrle +C عشان يقفل البرنامج بتاع السيرفر من غير ما يطلع من المسار بتاعه أو تقفل ال CLI خالص وتفتح السيرفر بالطريقة اللي عملناها من شوية أنت حر المهم تقفل السيرفر وتفتحه تاني.



تشغيل سكريت CGI

هتروح فى المجلد cgi-bin وهتعمل فيه السكريت ده وتحفظه بأى اسم مثلاً test.py

```
#!python
print("Content-type: text/html\n\n")
print('<h1 style="color:red">','Hello CGI','</h1>')
```

- السطر الأول ده بيحدد مكان المفسر الـ interpreter location مهم جداً وده بيعرف الأباتشى إنه هيعمل execute للسكريت ده بأى برنامج أو interpreter.
- الـ interpreter location بتاع لغة البايثون على Linux بيكون بالشكل ده #!/usr/bin/python أو لو كانت بايثون متثبته فى مسار تانى هتحتاج تكتب المسار كله.

بالنسبة لويندوز كتبنا #!python عشان مسار الـ interpreter متخزن فى الـ environment variables ومعروف بالنسبة للسيرفر ولو مش موجود كنا ه نكتب المسار كله مثلاً موجود عندى على ويندوز 10 وبايثنون 3.7 بالشكل ده

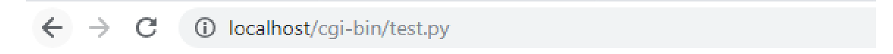
C:\Users\Mahmoud\AppData\Local\Programs\Python\Python36-32\python.exe

- السطر التانى مهم برده لأنه عبارة عن الهيدر اللى بيعرف السيرفر إن دى صفحة html وهتشتغل فى المتصفح ومن غيره أو من غير \n اللى فى الأخر السكريت مش هيشتغل.
- أخر حاجة الجملة اللى هنطبعها فى المتصفح.

بعد ما تحفظ السكريت تأكد إن الـ localhost شغال وبعدها هتكتب فى المتصفح



http://localhost/cgi-bin/test.py



Hello CGI

أى خطأ هيحصل معاك فى السطر الاول أو الهيدر أو الكود.. السيرفر هيجبلك صفحة Internal Error زى دى ومش هيقولك الخطأ فين فخلى بالك..

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at admin@example.com to inform them of the time this error occurred, and the actions you performed just before this error.

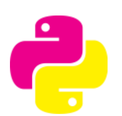
More information about this error may be available in the server error log.

و عشان تعرف أكتر عن استخدام CGI تقدر تشوف الشرح ده على موقع apache howto.

تطبيقات على صفحات CGI

استخدامك لأى لغة برمجة فى الويب هيكون ليها وظائف محدده وفى الغالب بيكون فى أستقبال الداتا من froms والتعامل مع قواعد البيانات database ورفع الملفات file upload والتعامل مع الـ cookies والـ sessions والحاجات دى.

تقنية CGI تعتبر من التقنيات البطيئة والقديمة فمش هنركز على تفاصيلها كتير لأن استخدام لغة البايثون فى الويب منتشر أكتر باستخدام الـ frameworks زى flask و Django واللى وهتكلم عن الاولى فى الفصل ده.



تطبيق Contact Form

هنعمل صفحة contct form وهي فورم html، المستخدم هيكتب بيانات في فورم ويعملها أرسال فبتتبعت بالإيميل وهتعتبر مثال كويس لطريقة دمج صفحة html في سكريبت بايثون والتعامل مع post request وقراءة قيم الـ inputs اللي في الـ form من خلال لغة البايثون.

- هتعمل صفحة html وتسميها contact.html وتقدر تحمل الكود بتاعها من GitHub.

Contact US

Name:

Enter Your Name

Email:

Enter email

Phone:

Enter Phone

Gendar:

Male

Message:

Submit

Download script contact.html from GitHub



- بعدين هتعمل سكريت contact.py وهتحط فيه الكود ده.

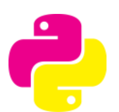
```
#!python
print ("Content-type: text/html\n")
import cgi,os
from mail import send

con={'server'  : 'smtp.mail.yahoo.com',
     'port'    : '587',
     'from'    : '...@yahoo.com',
     'pass'     : '*****',
     'to'      : '...@gmail.com',
     'subject' : 'CONTACT FORM'
    }
form={}

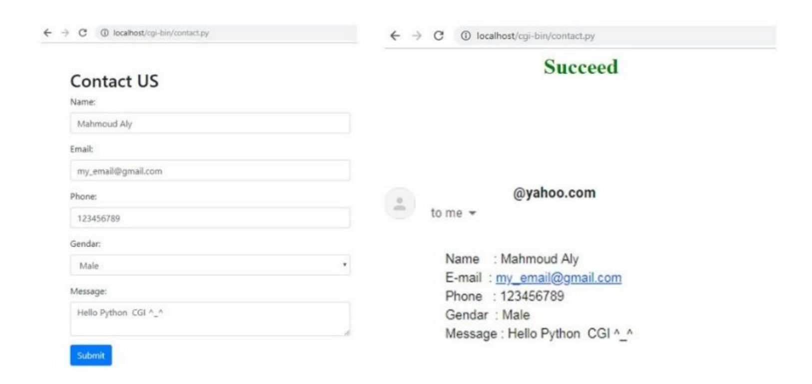
if os.environ['REQUEST_METHOD'] == 'POST':
   for key,val in dict(cgi.FieldStorage()).items():
       val=str(val)
       val=val[val.find(',')+3:len(val)-2]
       form[key]=val
   if send(con,form):
         print('<center><h1 style="color:green">Succeed<h1></center>')
   else:
         print('<center><h1 style="color:red">Failed<h1></center>')
else:
   html=open('contact.html','r').read()
   print(html)
```

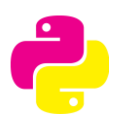
Download script contact.py from GitHub

- هنا أستخدمنا مكتبتين.. cgi ودى بتقرى القيم من الـ inputs فى الـ form ومكتبة os اللى لما المستخدم يعمل submit ويحصل post request فساعتها نبعت الإيميل.
- وعشان نبعت الإيميل أستخدمنا الموديول mail.py اللى عملناه فى الشابتر بتاع Packages ومكتبة stmplib.



- جملة الشرط if os.environ['REQUEST_METHOD'] == 'POST' بتتحقق لما المستخدم يملى البيانات فى الفورم ويدوس submit ويحصل post request.
- لو الشرط إتحقق دالة FieldStorage بترجع الفورم اللى بنعمل iteration عليها ونحاول نطلع القيم بتاعتها ونحطها فى dictionary اسمه form
- بعدين هنبعت الإيميل بالدالة send. اللى بتاخد أتنين dictionary وهم الحقول بتاعة الإيميل form وبيانات الإتصال con.
- جملة else فى أخر 3 سطور دى بتتحقق فى الحالة العاديه لما المستخدم يفتح الموقع على طول قبل ما يعمل submit وفى الحالة دى بنقرى الصفحة contact.html كملف نصى بدالة open ونعملها print وهى دى الى أنت بتشوفها لما تفتح الموقع على طول.
- طبعاً هتحط الملفات الثلاثة contact.py,mail.py.contact.html فى مجلد cgi-bin وهتفتح العنوان ده فى المتصفح http://localhost/cgi-bin/contact.py





وتقدر تحمل ملفات المشروع كله من هنا cgi-contact.zip

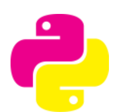
وده أخرى فى الكلام عن CGI لو عاوز تعرف أزاى ترفع الملفات file upload وتتعامل مع الـ cookies والـ session كمل فى الـ tutorial الجميلة دى على موقع tutorialspoint.

و لو عاوز تعرف أزاى تستضيف المواقع اللى بتعملها بالـ CGI على الأنترنت وأى حد يشوفها فأستنى شوية هتكلم عنها بعدين.

الفريمورك Framework

شوفت عشان نشغل صفحة CGI أضطرينا ننزل سيرفر Apache وعملناله configuration وكتبنا سكريت بايثون بطريقة معينة ده غير الأداء السىء للسيرفر اللى كل ما تعمل refresh للصفحة بيعمل execute للسكريت من جديد والموضوع ده بطئ.

فـ عملوا الـ frameworks وهى مجموعة من المكتبات لتسهيل عمل تطبيقات الويب بالبايثون.. وبيبقي معاها السيرفر بتاعها مش هحتاج لسيرفر الأباتشى وخصوصاً فى عملية التطوير أو التجريب debugging وهتقدر تتعامل مع قواعد البيانات وتعمل upload للملفات بشكل سهل وآمن وكمان بيعملك طريقة معينة فى التصميم design pattern بيفصلك فيها بين كود html وتصميم الموقع عن كود python وده بيخليك تعمل شغل منظم سهل الفهم والتعديل بالإضافة إنه بيقدر يدخل أكواد بايثون فى بين أكواد html بشكل سهل عن طريق الـ template enignes زى ما هتشوف دلوقتى.



فلاسك Falsk

flask من الفريم وركس البسيطة والمشهورة و تقدر تاخد فكرة عنها فى الموقع بتاعها وهتلاقى ليها documentation كويسة جداً ومن خلالها هنعمل تطبيقات ويب بشكل سهل ومنظم.

تقدر تثبت Flask عن طريق pip

```
pip install Flask
```

أول تطبيق على falsk

- هتعمل سكريت جديد وتسميه app.py أو تقدر تسميه أى اسم غير flask.py عشان ميتلخبطش مع اسم المكتبة.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return '<h1>Hello Flask</h1>'

app.run(debug=True)
```

سكريبتات flask مش بنشغلها فى الـ IDLE.هتفتح الـ command line فى المسار الموجود السكريت وتكتب اسمه app.py أو تدوس عليها مرتين وتشغله كـ executable file.



C:\Windows\py.exe

```
* Serving Flask app "flask-test1" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 679-601-415
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- لما شغلت السكريت هيشتغل السيرفر الخاص بالفريم ورك على العنوان ده http://127.0.0.1:5000 أنسخه فى المتصفح



Hello Flask

وهتظهر الصفحة قدامك..

نرجع للكود

- فى البداية بنعمل object من الكلاس Flask(__name__) وبياخد برامتر __name__ ده بيعرف للسيرفر أسم الأبلكيشن بتاعنا
- تانى حاجة دالة route بنحدد بيها الـ routes أو المسارات بتاعة الصفحات فى الأبلكيشن لما نفتحه فى المتصفح .
- مثلاً لما فتحت السيرفر فى المتصفح بالعنوان ده http://127.0.0.1:5000 ده اسمه الـ main route بتمثله العلامه / كبرامتر فى الدالة route('/').
- وعشان تفهم الموضوع أكتر.. أكتب فى المتصفح العلامة دى وبعدها أى حاجة زى كده http://127.0.0.1:5000/home مش هيلاقى المسار ده و هيقولك الصفحة مش موجودة.



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.



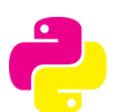
- طيب لو عاوز تحدد اكتر من route للصفحة الوحده بيكون بالشكل ده

```
@app.route('/')
@app.route('/home/')
@app.route('/index/')
def home():
    return '<h1>Hello Flask</h1>'
```

فلو كتبت فى المتصفح http://127.0.0.1:5000 أو http://127.0.0.1:5000/home أو http://127.0.0.1:5000/index هينفذ دالة home اللى بترجع جملة Hello Falsk اللى بتشوفها فى المتصفح .



- الدالة الموجود تحت الـ route هى اللى بتتنفذ لما تكتب عنوان الـ route فى المتصفح وتقدر تخلى اسمها أى اسم مش لازم يكون ليه علاقة باسم الـ route.
- أخر حاجة فى الكود app.run(debug=True) دالة run بتشغل السيرفر اللى بيعرض الموقع فى المتصفح والمتغير debug=True بيخليك شغال فى مود التجريب debugging بحيث إنك بتشغل السيرفر مرة وحده وتقدر تغير فى الكود فى السكريبت app.py وتعمل save وتفتح المتصفح تعمل refresh وتشوف التغيرات من غير ما تقفل السيرفر وتفتحه من جديد.



المتغيرات فى العنوان url variable

لو هتبعت متغير اسمه name وقيمته python عن طريق Get Request, فشكله بيكون زى كده http://127.0.0.1:5000/?name=python.. وفى فريم ورك فلاسك نقدر نزبط الصفحات بحيث تستقبل اسامى معينة من المتغيرات فقط

- فى الـ route هتكتب اسم المتغير او المتغيرات اللى عاوز تستقبلها بين العلامتين <>

```
@app.route('/page/<name>/')
@app.route('/page/<name>/<age>/')
def page(name="",age=""):
```

- مثلاً هنا عملنا route للصفحة اسمها page والمتغيرات اللى هتستقبلها أما متغير اسمه name أو متغير name و age.
- عشان تطلع المتغيرات من العنوان بتاع الصفحة وتاخدها فى سكريت الايثون بتمررها كبرامترات للدالة اللى تحت الـ route وطبعاً ممكن متغير يكون فاضى أو مش موجود فبنديه قيمة أفتراضية " " عشان ميحصلش خطأ.



هتعدل على سكريت app.py وحط فيه مسار جديد هنسميه about

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
@app.route('/home/')
@app.route('/index/')
def home():
    return '<h1>I`M Home Page</h1>'

@app.route('/about/')
@app.route('/about/<name>/')
@app.route('/about/<name>/<age>/')
def about(name="",age=""):
    return '<h1>My Name Is : %s<br>My Age Is : %s</h1>'%(name,age)

app.run(debug=True)
```

جرب تفتح الصفحة من العناوين دى about و about/Mahmoud و about/Mahmoud/23



- طبعاً مش المستخدم اللى بيكتب قيم المتغيرات بنفسه فى المتصحف ولكنها بتروح عن طريق GET Request بس بوريك طريقة استخراج المتغيرات من العنوان.



إضافة صفحات html

فى الـ frameworks بيظهر عندنا مصطلح جديد اسمه template وهو طريقة لدمج أكواد البايثون فى اكواد الـ html.

```
x='World'
print('<div>Hello %s</div>'%x)
```

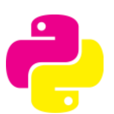
شوفنا وأحنا وشغالين فى سكريت CGI الموقع كان عبارة عن سكريت بايثون وكود html بعمله print وبطرق الـ string format اللى بتسمح بيها لغة البايثون بقدر أدخل متغيرات وقيم بتاعة لغة البايثون مع الـ tags بتاعة الـ html.

وشوفنا كمان فى مثال CGI لما حبيت أضيف صفحة كاملة عملتها فى ملف html وقريته بدالة file كنوع من التبسيط.

لكن مكتبة Flask بيستخدم template engine اسمه jinja بيحل المشكلة دى بيدمج المتغيرات بتاعة البايثون فى أكواد html وكمان بيدخل جمل الشرط التكرار فى أكواد html وهنشوف الموضوع ده مفيد ازاى والأهم من دول بيعزل صفحة html عن سكريت البايثون.

عشان تضيف صفحة html لمسار معين بتستخدم داله render_template.

```
from flask import Flask,render_template
@app.route('/page/')
def page():
  return render_template('page.html',msg='Hi ^_^')
```



- فى المثال ده لما تفتح المسار page/ هيعرضلك الصفحة page.html الموجودة فى الفولدر templates.. وهيمرر ليها متغير اسمه msg.
- لو عاوز أعرض المتغير بين أوكواد html فى صفحة page يكون بالشكل ده

```
<h1>Message From Python  : {{ msg }}</h1>
```

بتكتب اسم المتغير بين أقواس مجموعة {{variable_name}} بس بشرط تكون مررت المتغير ده كبرامتر فى الدالة render_template اللى بتربط بين صفحة html والـ route.

هنرجع لـ app بتاعنا.

- جنب السكريت app.py هتعمل مجلد اسمه templates وده اللى بيتحط فيه صفحات html وهتعمل فيه ملفين home.html وabout.html وشكل فولدر البرنامج هيكون كده.

```
- app.py
- templates
  -- home.html
  -- about.html
```

home.html

```
<h1>Hello ^_^</h1>
<p>Local Time  : {{time}} </p>
<p>Operating System : {{os}}   </p>
```

- دى صفحة html عاديه عاوزين نعرض فيها متغيرين جايين من البايثون وهم time و os عشان نعرض فى الصفحة الوقت ونظام التشغيل.



about.html

```
<h1>About Me</h1>
<p>My Name Is : {{name}}</p>
<p>My Age Is  : {{age}}</p>
```

- صفحة about هنعرض فيها متغيرين name و age

app.py

```
from flask import Flask,render_template
import time,platform
app = Flask(__name__)

@app.route('/')
@app.route('/home/')
@app.route('/index/')
def home():
   return render_template('home.html',time=time.ctime(),os=platform.system())

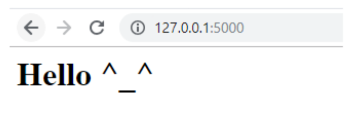
@app.route('/about/')
@app.route('/about/<name>/')
@app.route('/about/<name>/<age>/')
def about(name='',age=''):
   return render_template('about.html',name=name,age=age)

app.run(debug=True)
```

- تحت المسار home هنعرض الصفحة home.html بدالة render_template ونمرر ليها المتغيرين time و os.

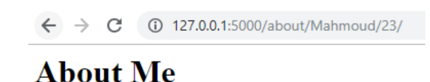
- نفس الكلام بالنسبة لصفحة about هنقرى الـ url varaibles اللى اسمهم name أو age ونعرضهم فى كود html فى صفحة about.html.





local time : Sun Sep 16 23:43:58 2018

Os Name : Windows



My Name Is : Mahmoud

My Age Is : 23

الشـرط والتكـرار فى الـ templates

قولنا مكتبة flask مستخدمة template engine اسمها jinja مش بس بيساعدك تدخل متغيرات البايثون فى الـ tags بتاعة html بس لكنه بيدعم عمل جمل الشرط والتكرار وحاجات تانية تقدر تشوفها فى الوثيقة من هنا docs

- إضافة متغيرات من لغة البايثون

```
<tag>   {{variable_name}}     </tag>
```

- جملة الشرط if وجملة التكرار for داخل الـ templates.

If condition	for loop
{%if condition1%} HTML CODE1 {%elif condition2%} HTML CODE2 {%else%} HTML CODE3 {%endif%}	{%for iteration_statement%} HTML CODE.. {%endfor%}



- لاحظ إن المتغيرات بتتحط بين أقواس {{ }} أما الشرط والتكرار بالشكل ده {% %} .
- خلاف الـ syntax بتاعة البايثون مفيش line indent هنا ومفيش علامة : تبدأ بيها البلوك.. وبتنهي البلوك بالجمل endif,endfor.

عشان نجرب الموضوع هنعدل على دالة about و هنعمل ليسته اسمها colors وفيها شوية أسامي للألوان ونمررها لصفحة about.html.

```
def about(name='',age=''):
   colors=['red','green','blue','orange']
   return render_template('about.html',name=name,
                                age=age,colors=colors)
```

about.html

```
<h1>About Me</h1>
{%if name!='' and age !=''%}
<p>My Name Is : {{name}}</p>
<p>My Age Is  : {{age}}</p>
{%else%}
<p>no information provided</p>
{%endif%}
<h1>Favorite Colors</h1>
<ul>
{% for color in colors %}
  <li style="color:{{color}}">{{color}}</li>
{% endfor %}
</ul>
```

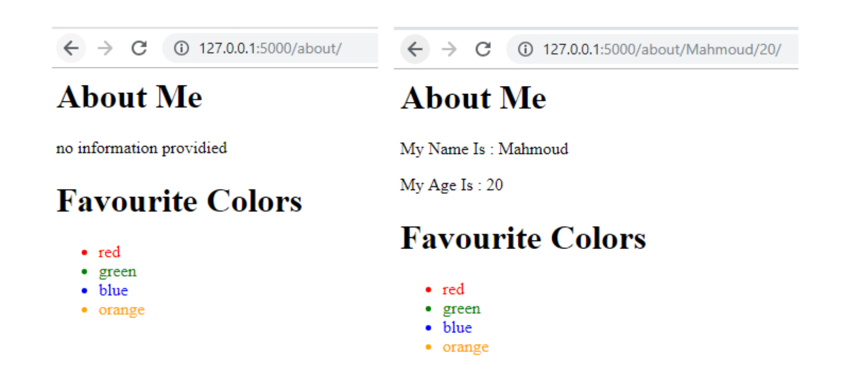
- في صفحة about في الاول في حالة إنه مكنش فيه قيمة للمتغيرين name و age في العنوان كانت النتيجة بتبقى كده

My Name Is :

My Age Is :



- بعد التعديل وإضافة جملة الشرط if.. إذا كان فية قيمة للمتغيرين name وage يطبعهم عادى ولو مفيش يطبع جملة no information provided.
- تالت متغير أو list وهو colors عملناله iteration فى جملة for وكونا بالتكرار عناصر الليسته <ul> بكود مختصر . ودى من مميزات استخدام التكرار فى html.



إضافة ملفات css و javascript

عشان تضيف ملفات css و javascript أو صور img فى الـ template بنحطهم فى فولدر بنسميه static فبيكون شكل البرنامج زى كده .

```
- app.py
- templates
  -- page.html
- static
  -- style.css
  -- script.js
```

ومش لازم تحط ملفات css و js جوة المجلد static على طول..

ممكن تحطهم فى مجلدات فرعية بالشكل ده static/js/script.js وstatic/css/style.css .



لما بتضيف الملفات لصفحة html بيكون بالشكل ده..

```
<script src="{{url_for('static', filename='script.js')}}"></script>
```

```
<link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
```

```
<img src="{{url_for('static', filename='img.jpg')}}">
```

- دالة url_for مهمة جداً لما تضيف ملف css او js أو img.. الدالة بتساعد السكريت يلاقى مكان الملف دى مهما كان مكان المشروع على السيرفر .
- كان ممكن تخلى المسار /static/file_name على طول .. بس ده ممكن يعملك مشاكل فى الإستضافة اللى هتنشر عليها الأبلكيشن لو حطيت المشروع فى مجلد فرعى عليها مش فى الروت المشكلة دى مش هتحس بيها على سيرفر فى debugging.
- ممكن برده تستخدم ملفات CDN أو روابط للملفات js,css على سيرفرات خارجية بشكل مباشر طبعاً.

هنتعلم دلوقتى أزاى نضيف مكتبة jquery داخل الـ template فى المشروع بتاعنا.

- فى مجلد templates هتعمل صفحة html جديده اسمها jquery.html .
- وفى الفولدر static هتحمل ملف jquery
- دلوقتى شكل فولدر الأبلكيشن هيكون كده..

```
- app.py
- templates
  -- home.html
  -- about.html
  -- jquery.html
- static
  -- jquery.min.js
```



jquery.html

```
<script src="{{url_for('static',filename='jquery.min.js')}}">
</script>

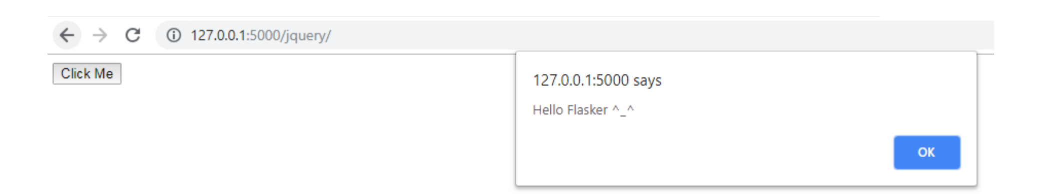
<script>
$(document).ready(function(){
    $("#btn").click(function(){
        alert('{{msg}}')
    });
});
</script>
<button id="btn">Click Me</button>
```

- دى صفحة html عاديه فيها كود جافاسكريت وعملت فيها إضافة لمكتبة jquery اللى موجودة فى الفولدر static.
- وفى كود jquery لما تدوس على الزرار اللى الـ id بتاعه btn هيعمل alert لمتغير جاى من البايثون msg..عشان تعرف إنك تقدر تدخل قيم من البايثون برده مع الجافا سكريت.

فى الملف app.py هتضيف route جديد للصفحة jquery.html

```
@app.route('/jquery/')
def jquery():
  msg="Hello Flasker ^_^"
  return render_template('jquery.html',msg=msg)
```

- هنا لما تفتح المسار /jquery هتعرض صفحة jquery.html ومررنا ليها المتغير msg.





قراءة المدخلات من صفحة html

عشان تاخد inputs من صفحة html لسكريت البايثون.. أما عن طريق إنك تعمل form أو تمرر الداتا فى http request بالـ ajax زى ما هنشوف

الفورم form

فى لغة html الـ from tag ده بيبقى فيه مجموعة inputs بندخل فيها البيانات اللى عاوزين نبعتها للسيرفر وبعدين بنعمل submit للفورم وبنحدد الـ method اللى هنبعت بيها الداتا أو الـ fields من الفورم للسيرفر وأما تكون POST/GET request.

وهناخد مثال يوضح طريقة أستلام الداتا من الفورم باننا موقع آله حاسبة بسيطة هيكون شكلها كده.

127.0.0.1:5000/form/

Falsk Calculator

10	+ ▾	50	500

Calc

- فى الفولدر templates هتعمل صفحة جديدة form.html وهتحط فيها الكود ده.

```
- app.py
- templates
  -- home.html
  -- about.html
  -- jquery.hyml
  -- form.html
```



Form.html

```
<h1>Falsk Calculator</h1>
<form action="" method="post">
   <input type="number" name='num1' value="{{num1}}" required>
   <select name="opr">
      <option>+</option>
      <option>-</option>
      <option>*</option>
      <option>/</option>
         <option>%</option>
      <option>**</option>
   </select>
   <input type="number" name='num2' value="{{num2}}" required>
   <label>{{result}}</label><br><br>
   <button type="submit">Calc</button>
</form>
```

- الفورم فيها 2 input المستخدم هيكتب فيهم الأرقام و select هيحدد منها العملية الحسابية واسمائهم num1,num2,opr لأننا هنطلع الـ values بتاعتهم من الفورم في تطبيق البايثون عن طريق اسمائهم.

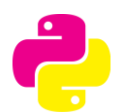
- لما نعمل submit للفورم الداتا هتتبعت عن طريق post request للعنوان اللى موجود في الـ action وأنا هنا سايبه فاضى..فالداتا هتروح لعنوان الصفحة اللى هنحدده في الـ route وهنسميه /form



هتيجى فى app.py وهضيف الـ route ده

```
from flask import request
@app.route('/form/', methods=['POST', 'GET'])
def form():
      result=num1=num2=''
      if request.method == 'POST':
          num1=request.form['num1']
          num2=request.form['num2']
          opr =request.form['opr']
          try:result=eval(num1+opr+num2)
          except:result='Math Error'
     return render_template('form.html',result=result,num1=num1,num2=num2)
```

- فى الاول عملنا import للكلاس request لأننا من خلاله هنتعرف إذا كان فيه post request جاى request.method وهنطلع منه القيم بتاعة الفورم request.from.
- بتطلع قيمة الحقل من الفورم عن طريق اسمه request.from[input_name].
- بعدين هنعمل execute بدالة eval للرقمين والمعامل عشان يجيب قيمتهم ويرجعها فى المتغير result اللى هيتبعت للصفحة تانى عشان يعرض النتيجة.



رفع الملفات Files upload

بتحتاج فى الموقع بتاعك إنك تعمل رفع للملفات او الصور زى المرفقات او صورة البروفايل أو الصور اللى فى المواضيع ..وطريقة الرفع ساهلة جداً فى Flask.

```
- app.py
- templates
- static
  -- uploads
```

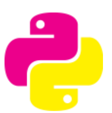
فى الاول هنعمل الفولدر اللى هيتحفظ فيه الملفات اللى هنرفعها

ممكن يكون فى أى مكان واى اسم بس لو عاوز ترفع صور وتستخدمها فى الـ templates فيفضل تخلى فولدر upload جوه static.

- فى فولدر templates هتعمل تمبلت جديد هتسميه upload.html

upload.html

```
<h1>File Upload</h1>
{%if done%}
<div>done uploading
   <strong>
<a target="_blank"
href="{{request.script_root}}/static/uploads/{{file}}">{{file}}</a>
   </strong>
   <p><a href="{{request.script_root}}/upload">back<a></p>
</div>
{%else%}
<form action="" method="post" enctype="multipart/form-data">
<input type="file" name="myfile" required>
<br><br>
<button type="submit" >Upload</button>
</form>
{%endif%}
```



- صفحة upload عبارة عن فورم فيها input نوعه file لما تحدد الملف اللي عاوز أرفعه وأدوس upload فيعمل submit للـ form ويبعت البيانات اللي فيها بـ post request ويستقبله بالدالة upload.
- في العناوين أستخدمنا request.script_root عشان بترجع مكان المشروع على السيرفر لما تحطه على الأستضافة.لو كان المشروع في مجلد فرعي script_root هيعرف السكريبت مكانه عشان ميغلطش في تحديد المسارات والعناوين.
- في tag الـ from فيه attribute enctype="multipart/form-data" ودي بتحدد نوع تشفير بيانات الفورم لما يتعملها submit والخاصية دة لازمة في حالة رفع الملفات.

في السكريبت app.py هتضيف روت جديد upload.

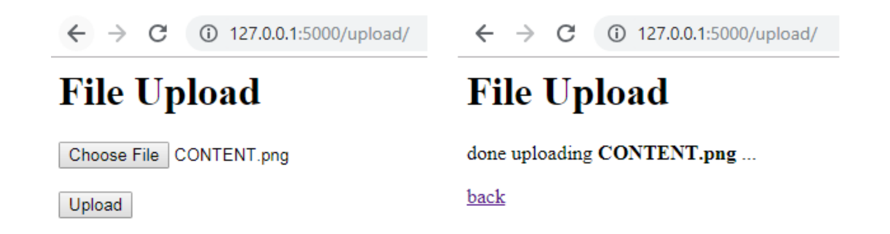
app.py

```
@app.route('/upload/',methods=['POST', 'GET'])
@app.route('/upload/',methods=['POST', 'GET'])
def upload():
    if request.method == 'POST':
        from werkzeug.utils import secure_filename
        f = request.files['myfile']
        f.save(app.root_path+'/static/uploads/'+secure_filename(f.filename))
        return render_template('upload.html',done=True,file=f.filename)
    return render_template('upload.html')
```

- في الأول طلعنا الملف اللي رفعناه request.files[input_file_name] في أوبجكت f اللي فيه اكتر من ميثود منهم save للحفظ والمتغير filename ده بيرجع اسم الملف.



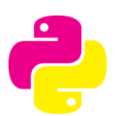
- دالة save بتاخد برامتر واحد وهو مسار حفظ الملف فى المجلد static/uploads + اسم الملف filename.
- app.root_path بترجع المسار الموجود عليه الأبلكيشن على السيرفر .. وعلى طول كده فى كل العناوين للصور والسكريتات فى كود html بتحط قبلها المتغير request.script_root وفى سكريت البايثون app.app_root.
- فى الوثيقة بتاعة فلاسك بيوصوا إنك تستخدم دالة secure_filename عشان تتأكد من إن اسم الملف آمن ويتنفع يتحفظ بيه على السيرفر بدل ما تستخدم f.filename على طول واللى بترجع اسم الملف برده.



الجلسات sessions

الـ session هى وسيلة لتخزين معلومة بشكل مؤقت لفترة ممكن تكون طول مدة تشغيل الموقع أو بتوصل لعدة لأيام.. المعلومة اللى بخزنهت المفروض إنى أقدر اوصلها من أى صفحة فى الموقع عشان تعرفنى بحاجة معينة.. مثلاً لما المستخدم يعمل تسجيل دخول فالمفروض أى صفحة أفتحها تعرف مين فاتح الصفحة وهل هو عضو مسجل دخول ولا ولا زائر عادى..

فبستخدم الـ sessions فى الموضوع ده.



ممكن تقولى أنا ممكن أخزن المعلومات دى فى متغيرات عاديه ليه أستخدم sessions ؟

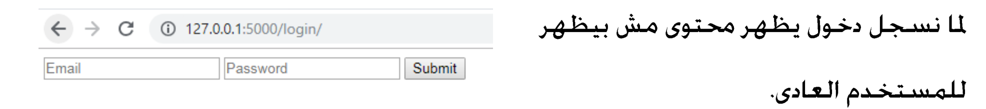
- عشان المتغيرات قيمتها بتروح لما تعمل refresh للصفحة ولكن الـ sessions بتفضل موجودة مهما عملت refresh ولأكتر من يوم كمان.

الـ session عبارة عن مكان بخزن فيه الداتا فتصورك ليه ممكن يكون ايه؟

- فى لغة الـ php الـ sessions بتتخزن فى ملف على السيرفر.. اما فى فريم ورك فلاسك بتتخزن فى الكوكيز cookies يعنى فى المتصفح بتاعك بس تفرق عن الكوكيز إن الفريم ورك بتعملها تشفير encryption بكود بنحدده فى الأبلكيشن بتاعنا زى ما هنشوف.

تسجيل الدخول

هناخد مثال على الـ session وهو عمل تسجيل دخول فى الموقع كـ admin بحيث إننا لما نسجل دخول يظهر محتوى مش بيظهر للمستخدم العادى.



- فهتروح فى فولدر templates وتعمل صفحة جديدة login.html وعبارة عن فورم فيها فيها 2 inputs واحد للإيميل والتانى للباسورد.

```
- app.py
- templates
  -- home.html
  -- about.html
  -- jquery.html
  -- form.html
  -- login.html
```



Login.html

```
{%if incorrect%}
<h3>Inccorect Email or Password! </h3>
{%endif%}
<form action="" method="post">
<input type="email" name='email' placeholder="Email" required>
<input type="password" name='password' placeholder="Password"
required>
<button type="submit">Submit</button>
</form>
```

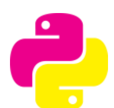
و هنعدل على صفحة home.html وهنحط فيها جزئين..

- Home
- Jquery
- Form
- Upload
- Login

- رسالة تظهر للمستخدم لما يكون مسجل دخول.
- وهنعمل list فيها عناوين الروابط لكل الصفحات اللى عملناها وفيه رابط login/logout.

home.html

```
<h1>Hello ^_^</h1>
{%if admin%}
<h4>You Are the SuperUser Of the Website</h4>
{%endif%}
<p>Local Time : {{time}} </p>
<p>Operating System : {{os}}   </p>
<ul>
<li><a href="{{request.script_root}}/home">Home</a></li>
<li><a href="{{request.script_root}}/about">About</a></li>
<li><a href="{{request.script_root}}/jquery">Jquery</a></li>
<li><a href="{{request.script_root}}/form">Form</a></li>
<li><a href="{{request.script_root}}/upload">Upload</a></li>
{%if admin%}
<li><a href="{{request.script_root}}/logout">Logout</a></li>
{%else%}
<li><a href="{{request.script_root}}/login">Login</a></li>
{%endif%}
</ul>
```



أخر حاجة هنعدل فى الأبلكيشن app.py

- هتعمل import للكلاس session و redirect.

```
from flask import Flask,render_template,request,session,redirect
```

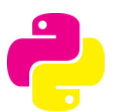
- تحت السطر (__app = Flask(__name هتحط الكود ده.

```
app.secret_key = 'd0dd342ebe25a8f6b7360e8c4183a5fc'
myLoginEmail='login_email@mail.com'
myLoginPass='123456'
```

- أول حاجة secret_key ده الكود اللى بشفر بيه القيم اللى هتتخزن فى الكوكيز على الجهاز أو الـ session.
- الـ key لازم يكون سرى ومميز..وتقدر تولده وتحدد طوله ونوع الصورة بتاعته بكلاس urandom فى موديول os.

```
>>> import os
>>> os.urandom(16).hex()
'd8527771bafd6a80496cb68fbbe22114'
```

- بعدين متغير الإيميل والباسورد اللى هتعمل بيهم تسجيل دخول..المفروض هتكون مخزنهم فى قاعدة بيانات والباسورد متشفر...



بعدين هنضيف الـ route بتاع تسجيل الدخول login

```
@app.route('/login/',methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        email     = request.form['email']
        password = request.form['password']
        if(email==myLoginEmail and password==myLoginPass):
            session['isAdmin']=True
            return redirect(request.script_root+'/')
        else:
            return render_template('login.html',incorrect=True)
    return render_template('login.html')
```

- فى البداية هتطلع القيم بتوع email و password من الفورم وتقارنهم بالمتغيرات اللى انت عرفتها أو الإيميل والباسورد اللى هتجيبهم من قاعدة البيانات.
- لو مطابقين يبقى كده تم تسجيل الدخول..فهيخزن فى الـ session متغير اسمه isAdmin وقيمته True ويعمل redirect لـ route "/" اللى هو الصفحة الرئيسية.
- لو شوفت الملف home.html عملنا فيه جملتين شرط بيتحققوا لو admin قيمته True و admin هيبقى True لما نعمل تسجيل دخول ونعمل redirect للمسار home.
- المهم دلوقتى عاوز نبعت قيمة المتغير admin لصفحة home.html.. فهتروح للـroute بتاع home وهتعدل عليه كده.



```
def is_admin():
    if 'isAdmin' in session:
        isAdmin=session['isAdmin']
        return isAdmin
    return False
@app.route('/')
@app.route('/home/')
@app.route('/index/')
def home():
  return render_template('home.html',time=time.ctime()
,os=platform.system(),admin=is_admin())
```

- ضفنا المتغير admin واللى المفروض قيمته تكون True لو كنت عملت تسجيل دخول وغير كده False.
- قيمة admin بترجعها الدالة is_admin اللى بتشوف الـ session هل فيها متغير اسمه isAdmin (اللى خزناه فى دالة login عند تسجيل الدخول) وبترجع قيمته True أو False
- المسار ده بيفتح صفحة home.html اللى عدلنا فى الكود بتاعها وخليناه لو admin=True هيظهر رسالة You Are the SuperUser Of the Website وفى الليسته تحت هيظهر خيار logout لعمل تسجيل خروج.



طبعاً logout ده route تسجيل الخروج ودالة logout متوقع إنها تخلى المتغير isAdmin اللى فى الـ session يساوى False وبعدين يحول للصفحة الرئيسية.

```
@app.route('/logout/')
def logout():
    session['isAdmin']=False
    return redirect(request.script_root+'/')
```

- ده شكل تسجيل الدخول على الموقع.

127.0.0.1:5000

Hello ^_^

local time : Wed Sep 19 02:54:22 2018

Os Name : Windows

- Home
- Jquery
- Form
- Upload
- Login

127.0.0.1:5000/login/

Inccorect Email Or Password !

mm@mail.com •••••• Submit

Home

127.0.0.1:5000/login/

login_email@mail.com •••••• Submit

Home

127.0.0.1:5000

Hello ^_^

You Are the SuperUser Of the Website

local time : Wed Sep 19 02:54:22 2018

Os Name : Windows

- Home
- Jquery
- Form
- Upload
- Logout

واخيراً تقدر تحمل ملفات الأبلكيشن كامل من هنا flask-test.zip



تطبيقات على فريم ورك فلاسك

دول 3 تطبيقات على فريم ورك Falsk بتشمل الأجزاء اللى شرحناها فى Falsk وبعض المكتبات اللى أخدناها فى الكتاب قبل كده.

Flask Contact From

نفس الـ contact from اللى عملناها بسكريت CGI حولتها لـ Falsk Application..

حاول تجرب الأتنين وتشوف الفرق فى السرعة ده غير تفوق فلاسك فى شكل وتنظيم الكود.

Contact US

Name:

Enter Your Name

Email:

Enter email

Phone:

Enter Phone

Gendar:

Male

Message:

Submit

لتحميل التطبيق من هنا flask-contact.zip



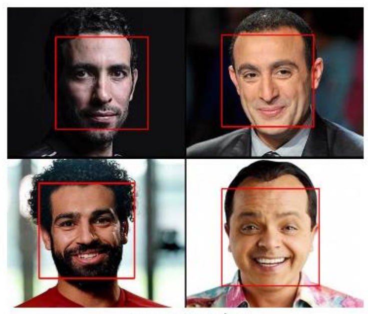
Flask OpenCV

ده تطبيق تانى بيدمج بين فريم ورك falsk ومكتبة openCV.. من خلالها بنعمل upload لأي صورة وبمكتبة OpenCV بنتعرف على وجوه الأشخاص فيها.

Flask/OpenCV Python Face Detection









input image

Faces Detected : 4

Flask/OpenCV Python Face Detection

Select Image

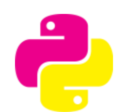


input image



Faces Detected : 2

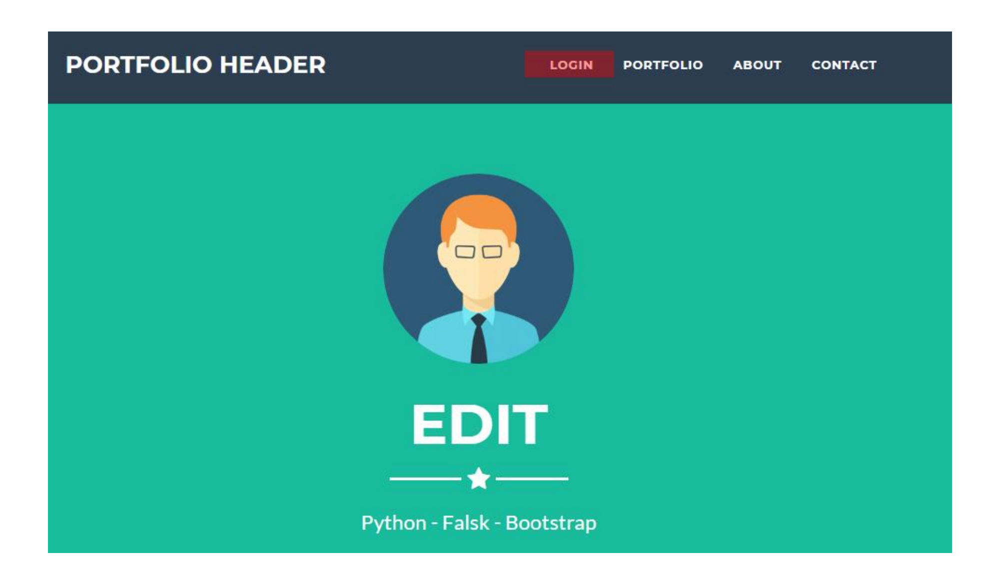
لتحميل التطبيق من هنا flask-opencv.zip

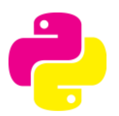


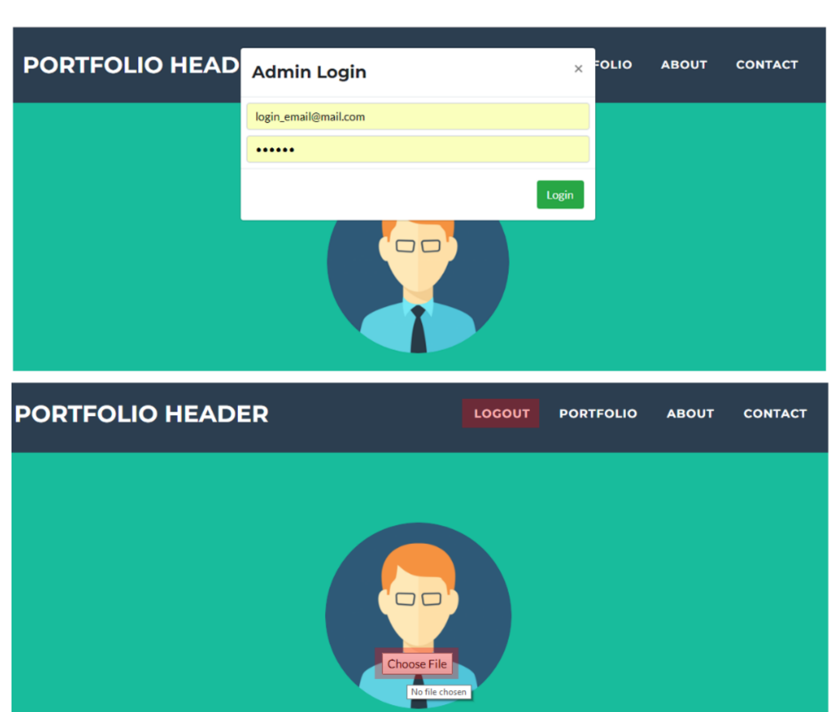
Editabel Portfolio

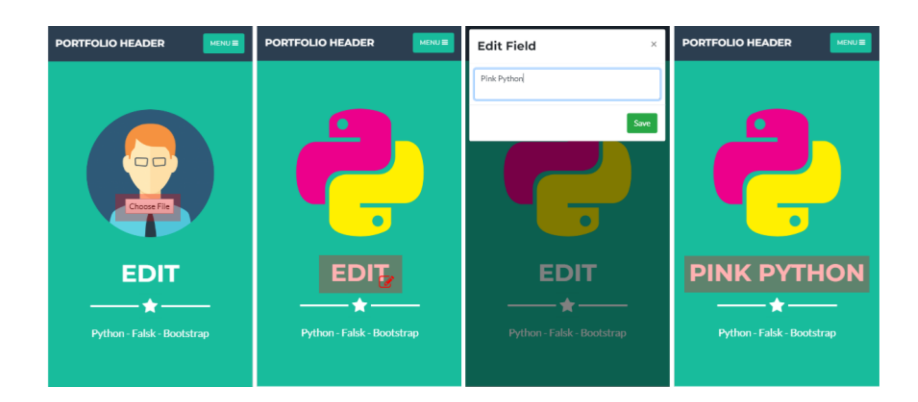
ده تـمبلت html مجانى علـى Github startbootstrap-freelancer لمعـرض أعـمال او Portfolio.

قدرنا بـ Falsk نعمل فيه نظام تسـجل دخـول وأمكـانية تعـديل كـل الحـقـول والعـنـاوين فى الصـفـحـة و إضـافة مـشـاريع لـلـ portfolio وتـغـيير صـورة البروفايل بشـكـل ديناميـكـى.

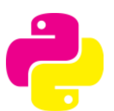








لتحميل التطبيق من هنا flask-portfolio.zip



أستضافة تطبيقات البايثون Python Hosting.

بالنسبة لإسكريتات CGI زى ما قولنا فأى إستضافة shared hosting عليها server Apache ومعموله configuration لتفعيل الأمر AddHandler cgi-script فتقدر تشغل عليها سكريتات بايثون بتقنية CGI.

تشغيل سكريتات بايثون على إستضافة جودادى.

إستضافة جودادى من أشهر الأستضافات اللى ينفع نشغل تطبيقات البايثون.

مثلاً إستضافة economy hosting أرخص باقة بتقدمها GoDaddy على سيرفر Linux مثبت عليه python 2 والجميل إنك تثبت عليها python3 و pip وتثبت الـ packages اللى هتحتاجها عن طريق ssh access من غير ما تحتاج تعمل ترقية للإستضافة أو تتشرك فى vps/cloud hosting.

بس تعالو نشغل أول سكريت بايثون CGI لينا على مفسر python2.

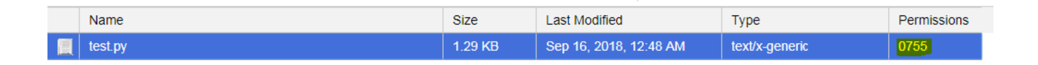
- هتروح فى المسار public_html/cgi_bin وهتعمل ملف جديد هتسميه test.py

```
#!/usr/bin/python
import platform
print ('Content-type: text/html\n')
print ('<h1>Hello From Shared Hosting ^_^</h1>')
print ('<h2>My operating system is %s<h2>'%platform.system())
print ('<h2>My python version is %s<h2>'%platform.python_version())
```

Download script test.py from GitHub



- أول سطر ده /usr/bin/python مسار python interpreter علـى Linux.
- طبعاً فاكر الهيدر Content-type: text/html\n
- بعدين بنطبع نوع نظام التشغيل وأصدار بايثون عن طريق موديول platform.

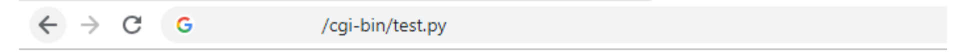


	Name	Size	Last Modified	Type	Permissions
	test.py	1.29 KB	Sep 16, 2018, 12:48 AM	text/x-generic	0755

- بعد ما ترفع او تحفظ الـ script لازم تخلى الـ permission بتاعه 755

  لو غير كده مش هيشتغل

- أى خطأ فى الكود أو الـ permission لما تفتح السكريت هيظهرلك internal error
- بعدين هتفتح السكريت www.domain_name.com/cgi_bin/test.py



Hello From Shared Hosting ^_^

My operating system is Linux

My python version is 2.6.6

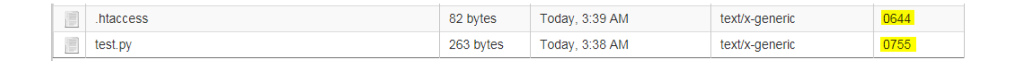
واحد هيسألنى هو لازم نحط سكريتات البايثون فى الفولدر الرخم ده cgi_bin مينفعش أى فولدر تانى؟ 🤔 الجميل إنه ينفع فى أى فولدر بس نعمل فيه ملف نسميه .htaccess وينفع نحط السكريت فى الروت public_html بتاع الأستضافة أو فى أى مجلد فرعى.



- دلوقتي في الروت public_html هتعمل فولدر وتقدر تسميه أي اسم طبعاً بس انت كالعاده هتعمله زيي وهتسميه python 😁ووهتنسخ فيه السكريت test.py اللي جربناه في الفولدر cgi_bin وأشتغل.
- جوة الفولدر python هنعمل ملف ونسميه .htaccess وتحط فيه السطرين دول.

```
AddHandler cgi-script .cgi.py
Options +ExecCGI
```

- لو الملف .htaccess مكنش ظاهر.. لازم لما تفتح الـ file manager من جديد و تختار show hidden files.
- خلي بالك من الـ permissions بتاعة الملفين.. test.py لازم يكون 755 و htaccess لازم يكون 644 غير ذلك مش هيشتغل السكريت



.htaccess	82 bytes	Today, 3:39 AM	text/x-generic	0644
test.py	263 bytes	Today, 3:38 AM	text/x-generic	0755

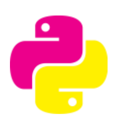
هتفتح العنوان بتاع سكريت البايثون your_domain/python/test.py



Hello From Shared Hosting ^_^

My operating system is Linux

My python version is 2.6.6



ولعلك تتسائل أنا ليه خافف الدومين بتاعى 🤔؟!

لأنك ربما تكون بتقرى الكتاب ده بعد سنه او ستنين من الآن وأكون آسفاً مجددتش الإستضافة والدومين وعشان جودادى مفتريه فى مصاريف التجديد فأحطلك روابط ولما تفتحها عندك متلاقيهاش😂.

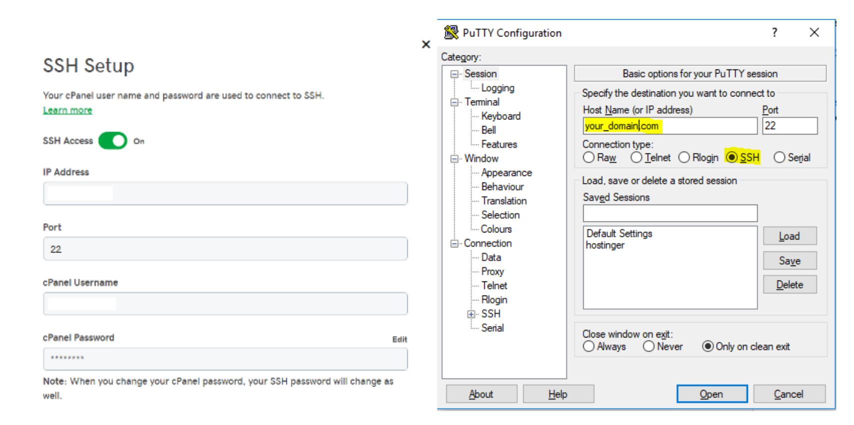
تثبيت python3

من الحاجات الجميلة على جودادى (economic host) إنك تقدر تثبت python3 وتثبت pip وتقدر تنزل كل الـ packages اللى أنت عايزهم زى youtube-dl و openCV والتقيله هنشغل تطبيقات Flask كمان 😂

طريقة التثبيت هنا على موقع جودادى how-to-install-and-configure-python-on-a-hosted-server بس فى الأول لازم تكون بتعرف تدخل للإستضافة عن طريق ssh وبتعرف تتعامل مع الـ terminal بتاع Linux .. ولو متعرفش فعدى الجزء ده وربنا يعوض عليك.

- فى الأول هتحمل أو لازم يكون عندك برنامج putty أو openssh أو أى ssh client ندخل بيه على الأستضافة وأنا هنا هشتغل على putty.
- فى موقع جودادى هتفتح cpanel>> hosting>> my account وهتفعل الـ ssh access وتعرف اليوزر والباسورد عشان تسجل بيهم فى putty.
- بعد كده هتفتح putty وتكتب الدومين بتاعك وبورت 22 وتختار ssh وتسجل دخول بالـ user والباسورد.





وهتنفذ الاوامر دى اللى شرحها فى الصفحة الجاية..

```
[~]$ cd public_html
[~/public_html]$ mkdir download
[~/public_html]$ cd download
[~/public_html/download]$ wget https://www.python.org/ftp/python/3.4.3/Python-
3.4.3.tgz
2018-09-18 19:03:31 (1.18 MB/s) - `Python-3.4.3.tgz' saved
[~/public_html/download]$ dir
./  ../  Python-3.4.3.tgz
[~/public_html/download]$ tar xvzf Python-3.4.3.tgz
[~/public_html/download]$ dir
./  ../  Python-3.4.3/  Python-3.4.3.tgz  Python-3.4.3.tgz.1  ]
[~/public_html/download]$ cd Python-3.4.3
[~/public_html/download/Python-3.4.3]$ ./configure --prefix=$HOME/.local

.
.
[~/public_html/download/Python-3.4.3]$ make
.
.
[~/public_html/download/Python-3.4.3]$ make install
.
.
```



- بعد ما تسجل دخول .. هتدخل بالـ terminal في المسار public_html

cd public_html

- هتعمل فولدر باسم اسم مثلاً download تحمل فيه لغة البايثون mkdir download

وهتدخل جوه الفولدر cd download

- دلوقتى هنحمل السورس ونعمله نعمله build ممكن تحمل أى إصدار بس اللى جربته ومعملش معايا مشاكل هو اللى الى في الشرح على الموقع مع إنه قديم شويه

.wget https://www.python.org/ftp/python/3.4.3/Python-3.4.3.tgz

- طبعاً تلاقيك مش عارف تعمل paste في البرنامج الغبي putty 😁.. هتدوس كليك يمين بالماوس مكان المؤشر بتاع الكتابه وهيعمل paste.

- هتفك ضغط الملف tar xvzf Python-3.4.3.tgz

- هتدخل في مسار المجلد اللى إتفك ضغطه cd Python-3.4.3

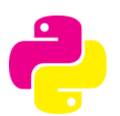
- هتنفذ الأمر ده اللى هيخلى بايثون يتثبت في مجلد اسمه /home/.local

./configure --prefix=$HOME/.local

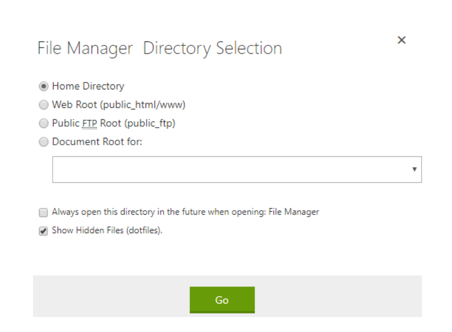
- تستنتى وبعدها تعمل make وبعدين make install وبكده تكون عملت build للبايثون في المسار /home/.local/bin بس لو كتبت python3 في الـ shell مش هيفتح بايثون لإننا لسه مضفناهاش في الـ environment variables.

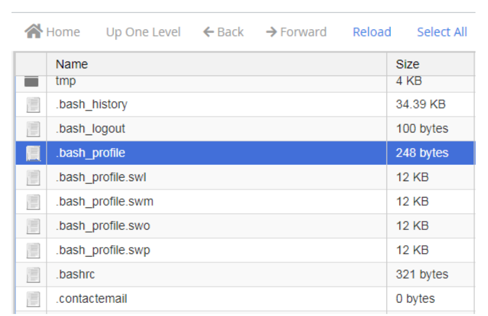
كده خلصت الاوامر اللى ناسخهالك من الـ terminal فوق فمتطلعش تبص تانى😁.

- عشان نضيف python3 في الـ environment variable لازم نعدل على ملف .bash_profile وده موجود في الـ home directory.



- في الشرح هو فتحه بالمحرر vim ومش عايزك تفتحه أحسن متعرفش تخرج منه 😂 وهنشتغل من الـ file manager أسهل.
- هتفتح الـ c panel وأنت وبتفتح الـ file manger أختار home directory وshow hidden files.
- في الـ home دور على الملف .bash_profile وأعمله edit.





- وفي أي حته في الملف هتضيف السطر ده

```
export PATH="$HOME/.local/bin:$PATH"
```

- وأخر حاجة هترجع للـ Terminal وتكتب الأمر ده عشان التغيير اللى عملته يتنفذ

source ~/.bash_profile

- وأخيراً اكتب python3 وهتشتغل معاك.. ولو مشتغلتش رستر الـ ssh client وشغله تانى😂.



```
[~/public_html]$ source ~/.bash_profile
[~/public_html]$ python3
Python 3.4.3 (default, Aug  6 2018, 12:43:02)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-18)] on Linux
>>> quit()
[~/public_html]$ pip3 --version
pip 6.0.8 (python 3.4)
[~/public_html]$ pip3 install -upgrade pip
```

كده قدرنا نثبت python3 و pip ونقدر نضيف أى package ونستخدمها فى تطبيقات الويب.

- مهم جداً إنك تعمل upgrade للـ pip عشان ميحصلش مشاكل فى تثبيت بعض الـ Packages.

سكريت CGI python3

هنروح فى الفولدر اللى عملناه فى public_html وأنا سميته python وعملنا فيه ملف htaccess. عشان نشغل فيه سكريتات CGI بس من خلال الـ python3.

- هتعمل سكريت جديد test2.py ومتنساش تخلى الـ persmissios بتاعته 755.

```
#!/home/<user_name_on_host>/.local/bin/python3
print ("Content-type: text/html\n")
import os,platform
print ('<h1>Hello From Shared Hosting ^_^</h1>')
print ('<h2>My operating system is %s<h2>'%platform.system())
print ('<h2>My python version is %s<h2>'%platform.python_version())
```

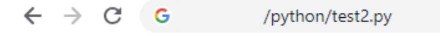
Download script test2.py from GitHub



- خلي بالك جداً أو سطر عملنا تغيير لمسار python interpreter عشان نستخدم python3 بدل python2
- وعشان تجيب مسار python3 من خلال الـ Terminal بالأمر which.

```
[~/public_html]$ which python
/usr/bin/python
[~/public_html]$ which python3
/home/my_usr_name/.local/bin/python3
```

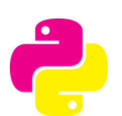
- هتعدل على المسار بتاع الـ interpreter وتجرب السكريبت اللي هنشغله على بايثون 3.



Hello From Shared Hosting ^_^

My operating system is Linux

My python version is 3.4.3



إستضافة تطبيقات فلاسك Deploying Flask App

نشر تطبيقات البايثون على الانترنت بيحتاج webserver بيدعم لغة البايثون بطريقة او باخرى .. زى سيرفر Apache قولنا بيتيح تشغيل سكريتات البايثون بأكتر من تقنية وموديول.. ومنها CGI command gate interface بأنه ياخد الكود و مسار الـ interpreter ويروح ينفذ الكود ويرجع النتيجة.. دى تعتبر الطريقة الأقدم بس فيها مشاكل فى الاداء.

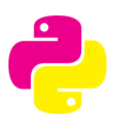
بعدين حاولوا يحسنوا الاداء بأنهم يدمجوا الـ Interpreter بتاع البايثون فى السيرفر و ظهرت تقنيات FastCGI و SCGI ..تقدر تعرف أكتر عنهم من هنا .webservers

وبعدين ظهرت تقنية اسمها WSGI web server gate way interface ودى بتخليك تقدر تشغل تطبيقات البايثون لأى فريم ورك بتدعم التقنية دى على أى سيرفر.. بمعنى إنك تقدر تشغل تطبيقات Flask على سيرفر Apache من غير ما تحتاج السيرفر بتاع Flask .

من الآخر هنقدر نشغل التطبيقات اللى عملناها بـ Flask على سيرفر الأباتشى اللى ثبتناه على الجهاز عندنا أو على أستضافة جودادى وهيساعدنا فى كده مكتبة اسمها wsgiref

تشغيل فلاسك على سيرفر أباتشي

- فى عملية النشر او deploying لتطبيقات Flask على أى سيرفر أو أستضافة لازم تشيل السطر الأخير اللى كنت تشوفه فى الأبلكشين app.run(debug=True) وده اللى بيشغل سيرفر فلاسك على الجهاز واحنا مش عاوزينه دلوقتى.



أول حاجة هنجرب على سيرفر أباتشى المتثبت عندنا على الجهاز .

- هتشغل سيرفر الأباتشى زى ما عرفنا فى الجزء بتاع CGI.
- هتروح المجلد cgi-bin وتنسخ فيه فولدر أى تطبيق من اللى عملناهم مثلاً flask-test.zip

```
-cgi-bin
 -flask-test
   -index.py
   -app.py
   -templates
   -static
```

- جوة الفولدر flask-test هتعمل سكريت جديد تقدر تسميه أى اسم مثلاً هخليه index.py وهو ده هيبقى الصفحة الرئيسية للتطبيق بتاعنا لما تفتحه فى المتصفح وهيكون بالعنوان http://localhost/cgi-bin/flask-test/index.py

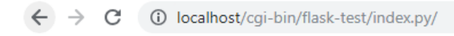
- هتفتح index.py وتحط فيه الكود ده.

```
#!python
from wsgiref.handlers import CGIHandler
from app import app
CGIHandler().run(app)
```

- طبعاً فاكر أن أول سطر ده interpreter location على ويندوز
- هتعمل import للموديول app.py اللى فيه كود Falsk
- والكلاس CGIHandler ده اللى هتشغل تطبيق Falsk باستخدام CGI.
- ونقول كمان..ياريت تتاكد إن app.py مفيهوش السطر اللى بيعمل run لسيرفر فلاسك.



- بعدين هتفتح العنوان ده http://localhost/cgi-bin/flask-test/index.py في المتصفح.



Hello ^_^

Local Time : Wed Oct 3 04:57:21 2018

Operating System : Windows

تشغيل تطبيقات فلاسك على أستضافة جودادى

طبعاً تقدر تحط التطبيق بتاعك فى أى مكان فى الأستضافة بس

لازم تضيف ملف .htaccess مع ملفات التطبيق.

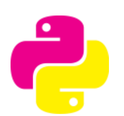
```
-flask-test
  -index.py
  -.htaccess
  -app.py
  -templates
  -static
```

index.py

```
#!/home/<user_name_on_host>/.local/bin/python3
from wsgiref.handlers import CGIHandler
from app import app
CGIHandler().run(app)
```

.htaccess

```
AddHandler cgi-script .cgi .pl .py
Options +ExecCGI
RewriteRule  ^/?$  index.py/
```



- تأكد إن الـ permissions بتاعة index.py معمولة 755 .
- السطر الاخير فى ملف .htaccess دى بيعمل rewrite للعنوان بتاع فولدر المشروع للصفحة index.py زى ما بيحصل فى صفحات php و html بحيث إنك لما تفتح الأبلكيشن بتكتب url/flask-test بدل ما تكتب العنوان كله url/flask-test/index.py

/python/flask-test/

Hello ^_^

Local Time : Tue Oct 2 20:14:43 2018

Operating System : Linux

- و لو عاوز تستضيف تطبيق flask-OpenCV هتحتاج تثبت مكتبة numpy و opencv-python بالـ pip من خلال الـ ssh client.

أستضافة pythonanywhere

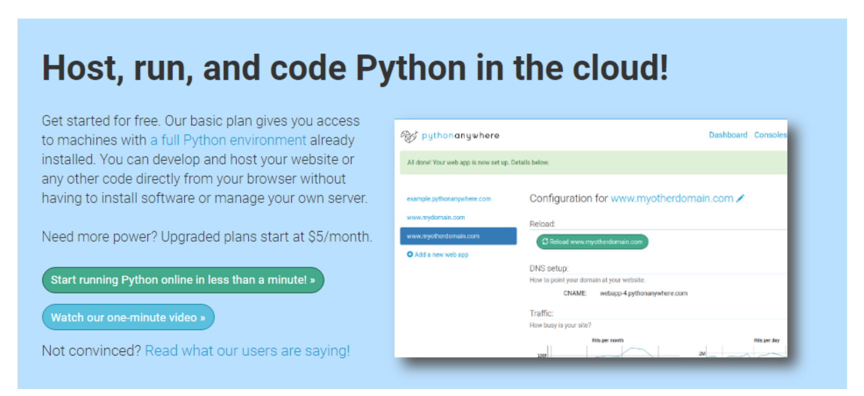
خلينا نتكلم كفنيين 😁 استخدام CGI لنشر تطبيقات البايثون مش فعال أوى و هتلاقى فيها بطء و مشاكل تانيه.

الحل أننا ننشر تطبيقاتنا على أستضافات من نوع cloud hosting .. و من إستضافات الـ cloud الجميلة و السهلة جداً واللى بتتيح عمل أكونت مجانى ورفع تطبيقات Flask أو أى فريم ورك تانية للبايثون أستضافة PythonAnywhere واللى أتكلمت عنها فى اول الكتاب وسجلنا عليها وأستخدمناها عشان نشتغل على البايثون أونلاين وقولت إننا

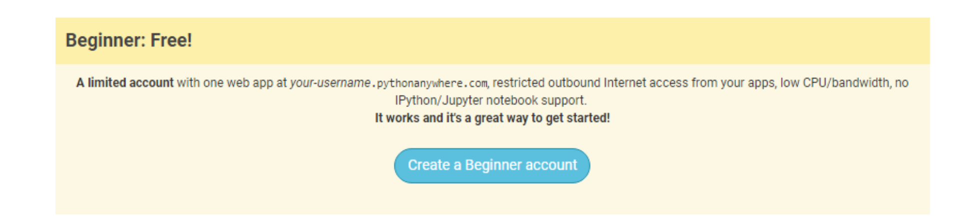


هنستخدمها لنشر تطبيقات البايثون على الإنترنت.. وفاتت الأيام وجه الوقت اللى هنستخدمها فيه 😁



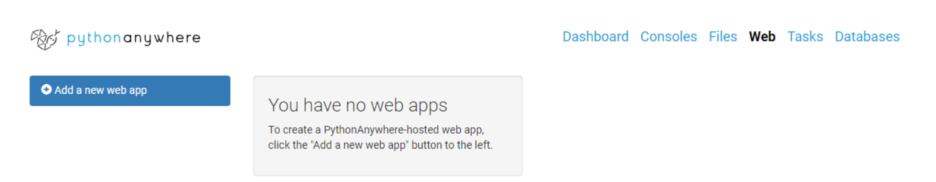


بأعتبار إنك مكنتش معانا فى أول الكتاب أو كسلت تسجل.. فتقدر دلوقتى تسجل فيها وتعمل أكونت مجانى بصلاحيات محدودة من هنا PythonAnywhere

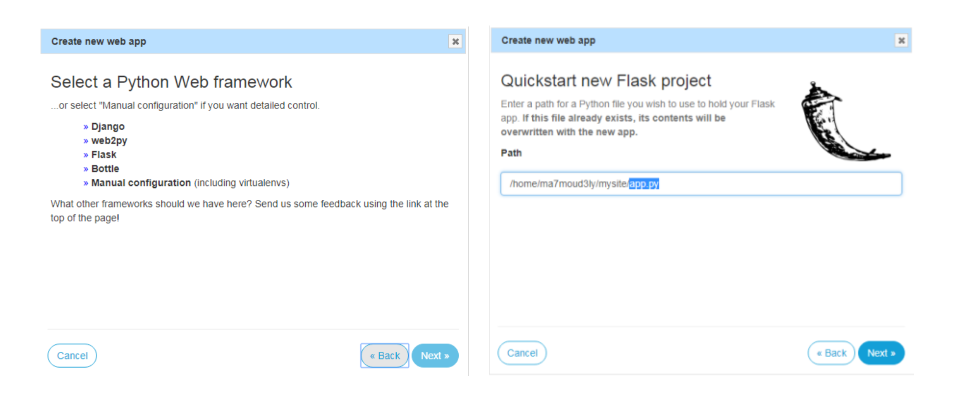


- بعد ما تسجل هتفتح الـ dashboard ومن فوق هتدوس على الرابط web وبعدين Add a new web app

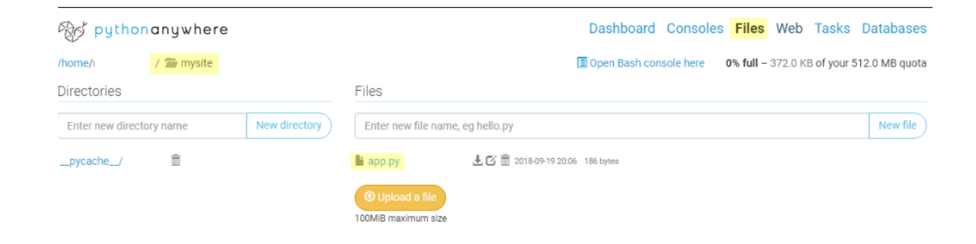




- هتختار الـ framework بتاعتنا وهي Flask وهتختار نسخة python اللي شغال عليها وهتكتب اسم السكريت الأساسي للتطبيق اللي كنا بنسميه app.py

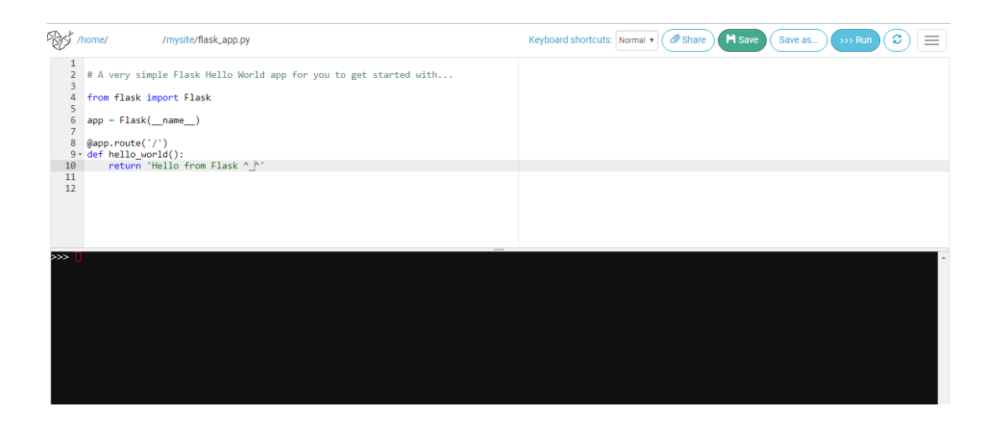


- كده انت أنشأت تطبيق جديد..
- بعد كده هتدوس على الرابط Files وعلى الشمال تدخل في الفولدر mysite وده اللي بتحط جواه ملفات الأبلكيشن بتاعك وهتلاقي فيه app.py





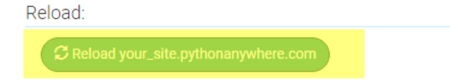
- تقدر تعمل edit للملف وتدخل للـ code editor اللى من خلاله تقدر تعمل run وتجرب الأكواد فى online interpreter.
- بعد ما تعمل تعديلات على السكريبت بتاعك و تدوس save وعشان التعديلات تظهر على الموقع لازم تعمل reload من الزرار اللى جنب Run او من صفحة web.



- وعشان تعرف الرابط بتاع موقع هتلاقيه فى صفحة web بعد جملة Configuration for.

Dashboard Consoles Files Web Tasks Databases

Configuration for your_site.pythonanywhere.com



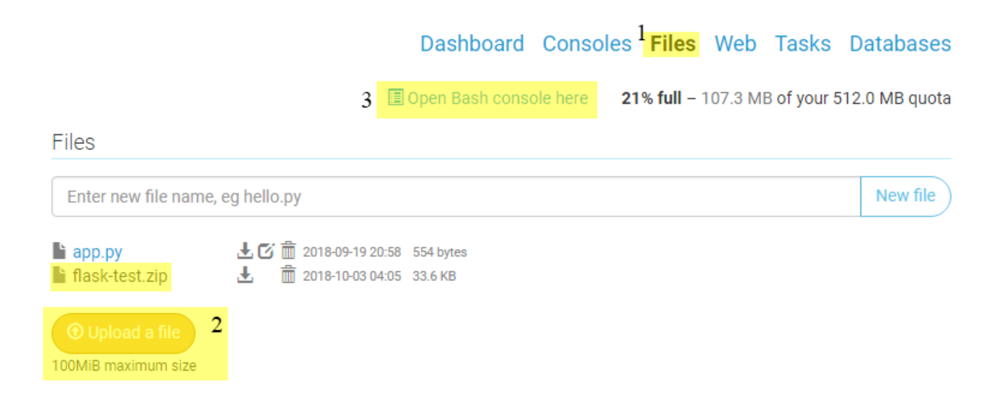
هتفتح الرابط فى المتصفح والموقع هيشتغل معاك..



رفع تطبيق Flask

عايزين نرفع التطبيق flask-test.zip على أستضافة PythonAnywhere

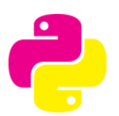
- فى الصفحة Files جوة المجلد mysite هتختار upload للملف التطبيق flask-test



- عشان نفك ضغط الملف مش هتلاقى أمر extract ولازم نعملها من Bash console اللى زى ssh access فى أستضافة GoDaddy.

- هتختار open Bash console here لما تفتح هتكتب الامر ده unzip file_name.zip

```
~/mysite $ dir
__pycache__   app.py   flask-test.zip
~/mysite $ unzip flask-test.zip
replace app.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: app.py
~/mysite $ exit
Exit Console closed.
```



- لما تفك الضغط لو قالك إن ملف app.py ده موجود عاوز تعمله replace هتكتب y وياريت فى الأخر تعمل الأمر exit.
- هترجع من الـ bash console وتروح فى المسار files/mysite هتلاقى الملفات كده



متنساش فى الآخر تروح صفحة web تعمل reload للموقع عشان التغيرات الجديدة تظهر ومن غيرها كأنك معملتش حاجة.

Dashboard Consoles Files Web Tasks Databases

Configuration for your_site.pythonanywhere.com

Reload:

Reload your_site.pythonanywhere.com

وبعدين أفتح الرابط فى المتصفح.

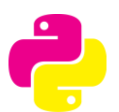
https://your_site.pythonanywhere.com

Hello ^_^

Local Time : Wed Oct 3 04:21:11 2018

Operating System : Linux

- لو ظهرت معاك أخطاء errors وانت وبتجرب التطبيقات هتروح فى صفحة web وتشوف الـ log files وخصوصاً error.log وحاول تشوف الاخطاء الموجودة وتصلحها.



Log files:

The first place to look if something goes wrong.

Access log:	your_site.pythonanywhere.com.access.log
Error log:	your_site.pythonanywhere.com.error.log
Server log:	your_site.pythonanywhere.com.server.log

Log files are periodically rotated. You can find old logs here: /var/log

عشان تثبت packages عن طريق pip هتفتح bash console وتستخدم إصدار pip حسب إصدار البايثون اللى شغالين بيه بصلاحيات user زى كده..

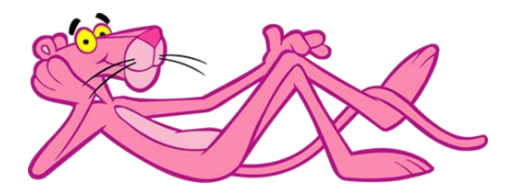
```
~ $ pip3.7 install youtube-dl --user
```

- مثلاً أحنا لما عملنا flask application فى الاول أخترنا إصدار بايثون 3.7 يبقى هنستخدم pip3.7 عشان ننزل بيه الـ Packages بدل ما ننزلها لإصدار بايثون تانى ولما نيجى نستخدمها فى تطبيقات Falsk منلاقيهاش.
- وبعد اسم الـ package بتكتب --user لأن دى أقصى صلاحياتك على الأستضافة المجانية ومن غيرها مش هيرضى يثبت الـ Package.

عموماً كفاية كده ويب .. وبعيداً عن أستضافة PythonAnywhere لو عاوز تعرف أكتر عن الاستضافات والسيرفرات اللى تقدر تنشر عليها تطبيقات flask فمن هنا deploying



الفصل 7 – التطبيقات الرسومية GUI



الفهرس



من بداية الكتاب وبنعمل تطبيقات console application اللى بتشتغل فى الـ command line الشاشة السوداء .. بعد كده أشتغلنا فى تطبيقات الويب web applications اللى بتشتغل فى المتصفح .. وجه الوقت أننا نتكلم شوية عن تطبيقات الواجهة الرسومية GUI graphical user interface.

وتطبيقات GUI هى أغلب التطبيقات اللى بتتعامل معاها بشكل يومى على الكمبيوتر والموبايل اللى فيها رسومات وزراير وصور وأداوت للكتابة وأدخال البيانات بعيداً عن شاشة الكونسول السودة اللى بتتعامل معاها فى الغالب بالكيبورد وكل اللى بتعمله إنك تكتب اوامر بس.

لبرمجة التطبيقات الرسومية بلغة البايثون فيه platforms ومكتبات كتيرة ومشهورة تقدر تعرفهم من هنا GuiProgramming.

لكن أختيارنا وقع على منصة اسمها Kivy

منصة Kivy

وهى مجموعة مكتبات مفتوحة المصدر ومن ميزاتها أنها cross platform يعنى نفس السكريت اللى بتكتبه هيشتغل معاك على نظام windows و Linux و android و ios.

من غير أى تعديلات وبتقدر تعمل تطبيقات desktop application و mobile application بنفس المنصة أو المكتبات.

دى طرق تثبيت وتحميل Kivy على أنظمة التشغيل المختلفة Download



- بس فى الاول تأكد إنك مثبت مكتبتين pygame و cython

```
pip install cython
pip install pygame
```

بعد كده بنثبت kivy ودى طريقة تثبيتها على ويندوز installation-windows.

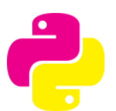
كتابة أول تطبيق

هتعمل سكريت جديد وتسميه main.py وحط فيه الكود ده..

```
from kivy.app import App
from kivy.uix.button import Button
class myApp(App):
    def build(self):
      return Button(text='Hello From Kivy',color=(1,0,0,1))
myApp().run()
```

Download script main.py

- فى الأول عملنا كلاس myApp ودى بترث الكلاس App وهى دى اللى هيبقى فيها أجزاء التطبيق بتاعنا.
- فى أخر سطر دالة run بتشغل التطبيق بتاعنا.. لما تعمل call للدالة run.
- أو دالة هتتنفذ دالة الكلاس myApp هى الدالة build ودى اللى بتعمل تصميم التطبيق فيها.



- دالة build بترجع الواجهة بتاعة التطبيق وهو زرار Button فيه مكتوب عليه Hello From Kivy عن طريق البرامتر text ولونه النص ده أحمر عن طريق البرامتر color.

- وبالنسبة للون color ففيه أكتر من طريقة لكتابة اللون فى kivy واللى أستخدمتها فى الزرار color = rgba(R,G,B,Alpha) .. والalpha دى الـ opacity او درجة وضوع اللون.

- ولاحظ إن قيم R,G,B بتتراوح بين 0 أسود و 1 أبيض على خلاف مكتبة OpenCV كانت 0 : 255.

بعد ما تعمل Run للتطبيق لو مفيش مشاكل هتظهر قدامك الشاشة دى.





تشغيل التطبيقات على الموبايل

قولنا تطبيقات Kivy بتشتغل على أنظمة تشغيل الموبايل زى android و ios.

بالنسبة لأجهزة PC فأى جهاز متثبت عليه Kivy هنقدر نشغل عليه السكريت.

بالنسبة لأجهزة الموبايل فعندك طريقتين لتشغيل تطبيقات kivy.

- ممكن تشغل التطبيق كـ script بتحطه فى فولدر اسمه kivy على ذاكرة الموبايل وتشغله عن طريق أبلكيشن تانى اسمه Kivy Launcher زى الـ Interpreter بتاع البايثون لكن على الموبايل.. والنظام ده تستخدمه فى مرحلة تجريب تطبيقاتك.
- المرحلة التانية وهى عمل Packaging للتطبيق عشان تقدر تنشره على play store او app store وتقدر تعرف أكتر عنها من هنا packaging-android packaging-ios.

تشغيل التطبيق عن طريق Kivy Launcher

عاوزين نجرب التطبيق اللى لسه عاملينه على الموبايل وشرحى هنا لنظام تشغيل اندرويد.

- أول حاجة هتحمل تطبيق Kivy Launcher من جوجال بلاي.
- هتروح لذاكرة الهاتف هتعمل فولدر اسمه kivy وده اللى هتحط فيه التطبيقات.

عشان نشغل التطبيق بتاعنا واللى هنسميه مثلاً test شكل الملفات هيكون كده

```
- kivy
  - test
    - main.app
    - android.txt
```

- هنعمل مجلد باسم التطبيق test جوة المجلد kivy.



- جوة test هيكون فيه ملفين.. السكريت main.py وملف الأعدادات android.txt

test/main.py

```
from kivy.app import App
from kivy.uix.button import Button
class myApp(App):
    def build(self):
      return Button(text='Hello From Kivy',color=(1,0,0,1))
myApp().run()
```

test/android.txt

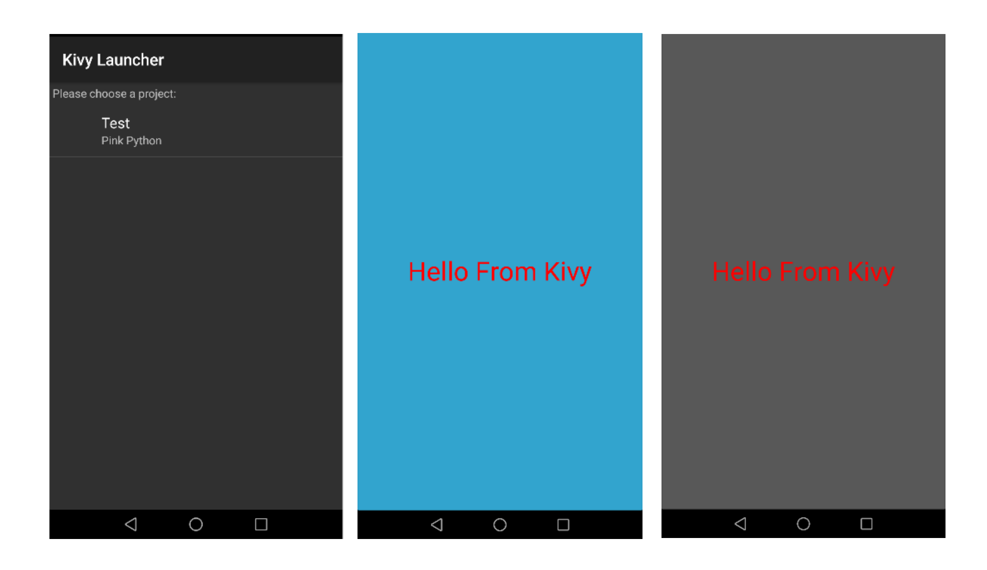
```
title=Test
author=Pink Python
orientation=portrait
```

- ملف android.test بتحط فيه أعدادت التطبيق زى الاسم title اللى هيظهر لما تيجى تفتحته فى Kivy Launcher وصاحب التطبيق وأخر حاجة orientation ودى وضع دوران الشاشة للتطبيق لم يشتغل هل معتدل portrait ولا مائل landscape.

- بعد ما تنسخ الفولدر test اللى جواه السكريت main.py والملف android.txt فى المجلد kivy.. هتروح تفتح تطبيق Kivy Launcher وهتلاقى التطبيق Test ظهر فيه فهتشغله

عشان تحمل التطبيق من هنا test.zip.





وده شكل التطبيق لما تشغله على الموبايل وتدوس على الزرار.

النافذة Window

(بالنسبة لتطبيقات الديسكتوب) لما شغلنا التطبيق الأول وبعيداً عن حجم الزرار اللي مالي الشاشة.. كان حجم النافذة قابلة للتكبير والتصغير و تقدر تعمله maximize و minimize .

لو عاوز تعمل حجم ثابت للنافذة..بتحدد الحجم ده عن طريق كلاس Config ومن خلالها برده تقدر تتحكم في حجم النافذة ولونها ومكان ظهورها في الشاشة.



- استدعاء الكلاس Config

```
from kivy.config import Config
```

- عشان تخلي حجم النافة ثابت

```
Config.set('graphics', 'resizable', False)
```

- عشان تحدد طول وعرض النافذة

```
Config.set('graphics', 'width','600')
Config.set('graphics', 'height','600')
```

- وضع fullscreen.. لو كان auto هيخلي حجم النافذة ملىء الشاشة.. لو كان 1 هيخلي حجم النافذة بأبعاد width وheight.. لو كان fake هيخلي النافذة من غير borders.

```
Config.set('graphics', 'fullscreen', 'auto')
Config.set('graphics', 'fullscreen', '1')
Config.set('graphics', 'fullscreen', 'fake')
```

- تحديد مكان ظهور النافذة بأحداثيات top و left.

```
Config.set('graphics', 'position', 'custom')
Config.set('graphics', 'top', '0')
Config.set('graphics', 'left', '0')
```



- عشان تحدد لون النافذة (لونها أسود بشكل أفتراضى)

```
from kivy.core.window import Window
Window.clearcolor = (R,G,B,Alpha)
```

- ده مثال بيظهر نافذة لونها أبيض من غير border حجمها 400 * 400 فى المكان (0,0) أعلى يسار الشاشة

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.config import Config
from kivy.core.window import Window
Window.clearcolor = (1, 1, 1, 1)
Config.set('graphics', 'resizable', False)
Config.set('graphics', 'width','400')
Config.set('graphics', 'height','400')
Config.set('graphics', 'position', 'custom')
Config.set('graphics', 'top', '0')
Config.set('graphics', 'left', '0')

class myApp(App):
    def build(self):
      return Button(text='Hello From Kivy',color=(1,0,0,1))
myApp().run()
```

Download script main.py

حبيت بس أكلمك عن إعدادات الـ window أو الفورم فى الأول عشان لو جاى من الفيجوال ستوديو متقولش انى حارمك من حاجة 😁

ولو عاوز تعرف أكتر عن كلاس Config من هنا api-kivy.config

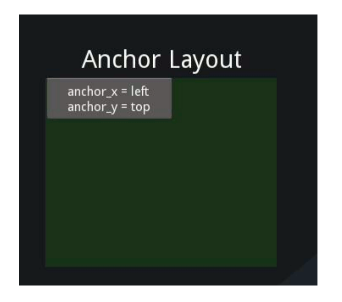


التصميم Layouts

عشان تعمل التصميم بتاعك محتاج يكون داخل container تحط جواه عناصر الصفحة أو الـ widgets زى Button وTexbox والـ tools اللى بتتعامل معاها.. ومحتاج برده تحدد أماكن العناصر دى بطريقة تسمح أنها تخلى التصميم متناسق على جميع أحجام الشاشات والموضوع بيفرق جداً فى التطبيقات اللى بتشغلها على الموبايل.

عشان تتحكم فى الموضوع ده kivy عملتلك مجموعة layouts هنتكلم عن بعضهم وتقدر تتعرف عليهم من هنا layouts

Anchorlayout



anchorlayout دى بتخليك تحدد أماكن العناصر اللى جواها عن طريق تحديد المحاذاة على المستوى الأفقى بأن العنصر يبقى left أو right او center وعن طريق المحاذة الرأسية top و bottom و center.

- استدعاء الكلاس AnchorLayout

```
from kivy.uix.anchorlayout import AnchorLayout
```

- عمل object من الكلاس AnchorLayout وتحديد محاذاة العناصر بداخلها.

```
layout=AnchorLayout(anchor_x='center or left or right',
                    anchor_y='center or top or bottom'
                    )
```



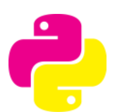
ده مثال على استخدام AnchorLayout لمحاذاة زرار Button في المنتصف

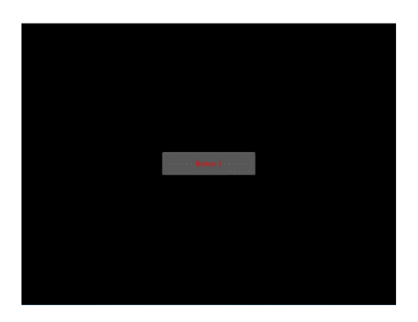
```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.anchorlayout import AnchorLayout
class myApp(App):
    def build(self):
        layout=AnchorLayout(anchor_x='center',anchor_y='center')
        btn=Button(text='Button 1',

color=(1,0,0,1),size=(200,50),size_hint=(None,None))
        layout.add_widget(btn)
        return layout
myApp().run()
```

Download script main.py , for android anchor-layout.zip

- في الكود ده عملنا layout من نوع AnchorLayout وحددنا المحاذاة الأفقية والرأسية للعناصر اللي هتكون فيها anchor_x=center و anchor_y=center
- عملنا زرار اسمه btn وهنتكلم فيما بعد هنتكلم عن خصائصه فمتشغلش نفسك دلوقتي غير إن ده زرار Button.
- و باستخدام الميثود add_widget أضفنا الزرار btn لـ layout.

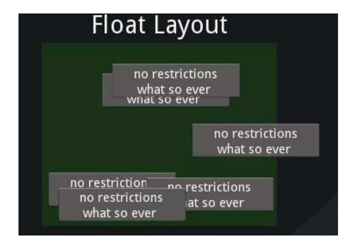




بعد ما تشغل ده شكل الزراز فى منتصف

.anchorlayout

FloatLayout



floatlayout من خلالها تقدر تحدد الـ position

بتاع الـ widgets بحيث إنهم يكونوا فى أى مكان

فى الصفحة عن طريق تحديد الأحداثى لكل عنصر.

- استدعاء FloatLayout

```
from kivy.uix.floatlayout import FloatLayout
```

- عمل object من FloatLayout

```
layout=FloatLayout()
```

- محاذاة العناصر جوة FloatLayout يتم عن الخاصية pos للعنصر نفسه اللى بتحدد الأحداثيات x,y ليه.. مثلاً هضيف زرار وعاوز الأحداثى بتاعه (50,50)

```
Button(text='Button', pos=(50,50))
```



مثال على استخدام FlaotLayout

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.floatlayout import FloatLayout
from kivy.config import Config
Config.set('graphics', 'resizable', False)
Config.set('graphics', 'width','400')
Config.set('graphics', 'height','400')

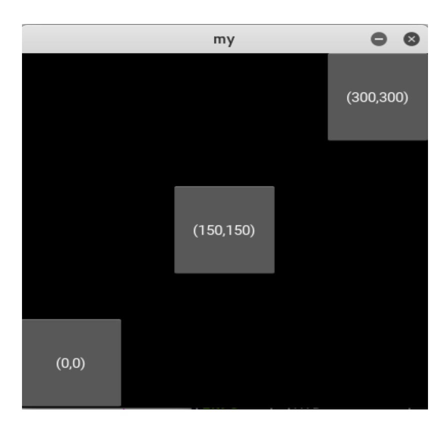
class myApp(App):
    def build(self):
        layout=FloatLayout()
        btn1=Button(
            text='(0,0)',size=(100,100),size_hint=(None,None),
            pos=(0,0)
          )
        btn2=Button(
            text='(150,150)',size=(100,100),size_hint=(None,None),
            pos=(150,150)
          )
        btn3=Button(
            text='(300,300)',size=(100,100),size_hint=(None,None),
            pos=(300,300)
          )
        layout.add_widget(btn1)
        layout.add_widget(btn2)
        layout.add_widget(btn3)
        return layout

myApp().run()
```

Download script main.py , for android float-layout.zip



شكل التطبيق لما تشغله



- فى الأول عملنا object من الكلاس FlaotLayout اسمه layout
- وبعدين عملنا 3 أزرار btn1,btn2,btn3 وحطناهم جوه layout بالـ add_widget
- حددت مكان كل زرار عن طريق البرامتر pos(x,y) مع مراعاة إن الأحداثى (0,0) ده فى أسفل يسار الـlayout مش فى أعلى اليسار زى ما هو متوقع.

ممكن يكون الأسلوب ده مش عاجبك ومش عاوز تتعامل بالأحداثى كأرقام عشان كده المكتبة عملتلك نظام يعتمد على النسبة المئوية للإحداثيات عن طريق الخاصية pos_hint

```
Button(text='Button', pos_hint={'x':0.5,'y':0.4})
```

- قيم x,y بتتراوح ما بين 0 : 1.. مثلاً هنا {pos_hint={'x':0.5,'y':0.4 الزرار ده مكانه هيبقى على بعد 50٪ من العرض و 40٪ من الطول بتوع الـ layout.



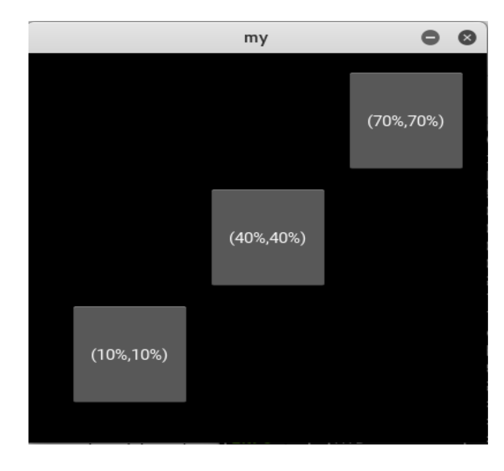
لو هنعدل فى المثال اللى فات شكل الزراير هيكون كده..

```
btn1=Button(
     text='(10%,10%)',size=(100,100),size_hint=(None,None),
     pos_hint={'x':0.1,'y':0.1}
  )
btn2=Button(
   text='(40%,40%)',size=(100,100),size_hint=(None,None),
     pos_hint={'x':0.4,'y':0.4}
  )
btn3=Button(
     text='(70%,70%)',size=(100,100),size_hint=(None,None),
     pos_hint={'x':0.7,'y':0.7}
  )
```

Download script main.py , for android float-layout2.zip

نظام النسبة المئوية هيكون أفضل لأنك صعب تتحكم فى الأحداثيات على شاشات الموبايل

حتى جرب الأبلكيشن ده واللى قبله على الموبايل وشوف الفرق.





BoxLayout

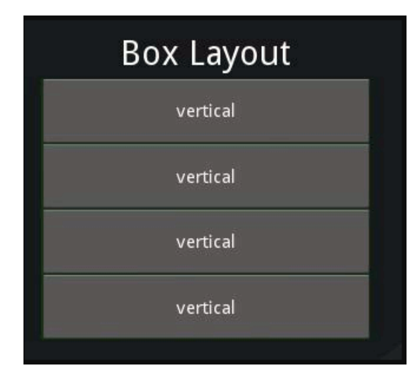
لو عاوز متشغلش نفسك بال position والـ anchor والـ size يبقى تتعامل مع BoxLayout واللى تعتبر الأفضل فى تنسيق وتنظيم العناصر داخل التطبيق.

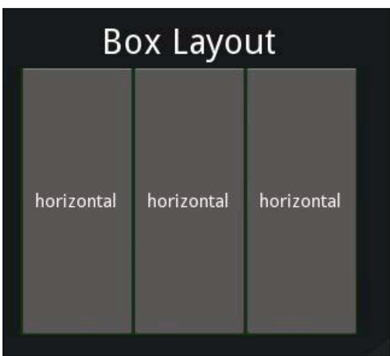
- استدعاء الكلاس BoxLayout

```
from kivy.uix.boxlayout import BoxLayout
```

- عمل object من BoxLayout

```
layout=BoxLayout(orientation='horizontal')
layout=BoxLayout(orientation='vertical')
```





- الكلاس بتاخد برامتر مهم وهو orientation وده بيحدد إصطفاف العناصر جنب بعض بشكل أفقى ولا رأسى.

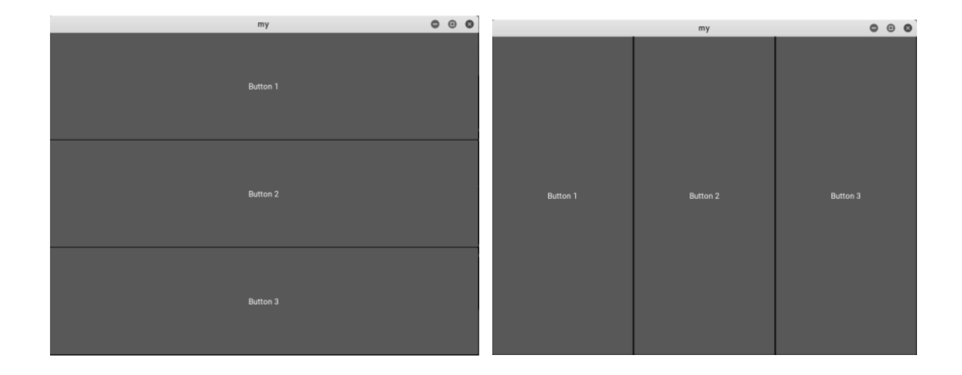


- في الـ BoxLayout مش هتحتاج تحدد الأبعاد بتاعة العناصر لأنها هتتقسم بالتساوى بناءاً على أبعاد الـ layout ونوع الـorientation والمثال هيوضحلك.

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
class myApp(App):
    def build(self):
        layout = BoxLayout(orientation='horizontal')
        btn1 = Button(text='Button 1')
        btn2 = Button(text='Button 2')
        btn3 = Button(text='Button 3')
        layout.add_widget(btn1)
        layout.add_widget(btn2)
        layout.add_widget(btn3)
        return layout
myApp().run()
```

Download script main.py , for android box-layout.zip

لو عاوز الزراير تظهر فوق بعض هتغير الـ orientation وتخليها vertical.





- تقدر تعمل اكتر من boxlayout جوه boxlayout عشان تعمل عناصر مصطفة بشكل أفقى وعناصر بشكل رأسى مع بعض.

- الـ box layout بيحدد العرض الأرتفاع للعناصر جواه بناءاً على نواع الـ orientation وبيقسم العناصر بالتساوى.

لو عاوزين نحدد عرض او أرتفاع أى عنصر داخل الـ BoxLayout بنستخدم خاصية size_hint=(x,y).. حيث x,y قيمتهم بين 0:1 وتعبر عن نسبة مئوية.

مثلاً عاوز أعمل زارار داخل الـ BoxLayout أرتفاعه 100٪ من أرتفاعها وعرضه 50٪ من عرضها هيكون كده

```
Button(text='Button', size_hint=(0.5 , 1))
```

وده مثال هيوضحلك.

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout

class myApp(App):
    def build(self):
        layout = BoxLayout(orientation='vertical')

        layout1 = BoxLayout(orientation='horizontal',
        size_hint=(1,0.1))
        btn11 = Button(text='Button 11',size_hint=(0.8,1))
        btn12 = Button(text='Button 12',size_hint=(0.1,1))
        btn13 = Button(text='Button 13',size_hint=(0.1,1))
        layout1.add_widget(btn11)
        layout1.add_widget(btn12)
```



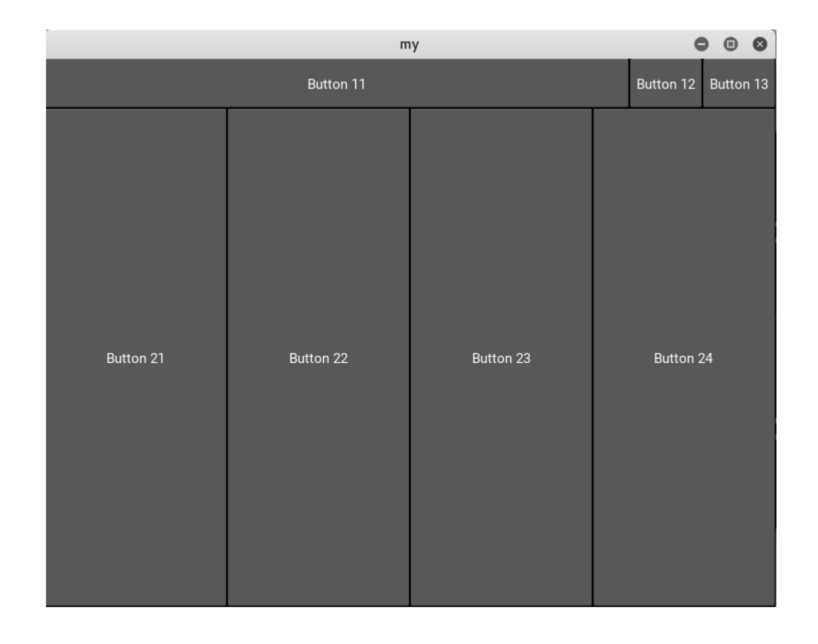
```
        layout1.add_widget(btn13)

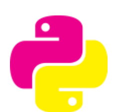
        layout2 = BoxLayout(orientation='horizontal')
        btn21 = Button(text='Button 21')
        btn22 = Button(text='Button 22')
        btn23 = Button(text='Button 23')
        btn24 = Button(text='Button 24')
        layout2.add_widget(btn21)
        layout2.add_widget(btn22)
        layout2.add_widget(btn23)
        layout2.add_widget(btn24)

        layout.add_widget(layout1)
        layout.add_widget(layout2)
        return layout

myApp().run()
```

Download script main.py , for android box-layout2.zip





العناصر Widgets

الـ widgets هى العناصر التقليدية المكونة للواجهة الرسومية زى Label, Button, CheckBox, Image, Slider, Progress Bar, Text Input, Toggle button, Switch, Video

الازرار Button

مش عاوز أوصف الـ widgets عشان ما يبقاش وصف وأهبل 😂 بس أنت المفروض عارف إن

الزرار ده عنصر بتدوس عليها عشان ينفذ حاجة معينة لما تدوس عليه press او تلمسه touch أو تسيبه release دى تسمى events احداث بتحصل للزرار و ينفع تتربط بدوال عشان تنفذ حاجات معينة عند حدوثها.

- عشان تعمل import للكلاس Button

```
from kivy.uix.button import Button
```

- عشان تعمل object من الكلاس Button

```
button=Button()
```

- خصائص الزرار Button Attributes



text	النص داخل الزرار
font_size	حجم النص
Color	لون النص
background_color	خلفية أو لون الزرار
id	رقم او اسم مميز للعنصر

- الخصائص دى ممكن تمررهم كبرامتر فى الـ constructor بتاع الكلاس Button او كـ attributes للـ object اللى بتعمله منها اللى بتعمله من الكلاس.

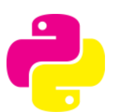
```
btn=Button(text='Hello')
btn.text='Hello World'
```

- عشان تحدد حجم الزرار فى طريقتين..أما بتحدد الحجم بالبيكسل عن طريق الخاصيتين size , size_hint

```
size=(width,height)
size_hint=(None,None)
```

- أو تستخدم الخاصية size_hint بس عشان تحدد حجم الزرار بالنسبة المئوية من أبعاد الـcontainer أو الـ layout الذى بداخله الزرار.

```
size_hint=(width%,height%)
```



- النسبة بترراوح بين 0:1 .. مثلاً الزرار صاحب الخاصية دى size_hint=(0.4,1) عرضه 04٪ من عرض الـcontainer وأرتفاعه 100٪ يعنى نفس الأرتفاع.

الأحداث Events

الزرار ليه 3 أحداث مهمين وهم on_press لما تضغط عليه on_release لما تسيبه on_touch لما تلمسه على شاشات التاتش..و الحدث event بياخد function بينفذها لما يحصل الحدث ده.

```
def press(instance):
  print('pressed..')
btn=Button(on_press=press)
```

- هنا عملنا زرار btn لما تدوس عليه هيحصل الـ event بتاع on_press اللى هينفذ الدالة press وهيمرر ليها برامتر انا سميته instance او تقدر تسميه اى اسم. ده بيبعت الـ object بتاع الزرار للدالة press عشان لو عاوز تطلع منه أى خصائص متعلقة بالزرار زى النص أو id بيكون كده instance.text أو instance.id

```
from kivy.app import App
from kivy.uix.button import Button
class myApp(App):
    i=0
    def press(self,instance):
        instance.text=str(self.i)
        self.i+=1
    def build(self):
        layout=BoxLayout()
```

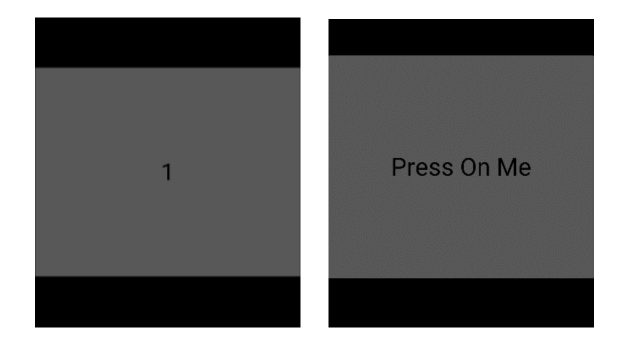


```
        btn=Button(
              text="Press On Me",
             color=(0,0,0,1), #black
             background_color=(1,1,1,1), #white
             font_size=100,
             size_hint=(1,0.5),
             pos_hint={'y':0.25},
             on_press=self.press,
            )
        layout.add_widget(btn)
        return layout
myApp().run()
```

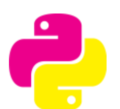
Download script main.py , for android button.zip

السكريت ده عبارة عن button عرضه ٪100 من عرض الشاشة وأرتفاعه ٪50

فيه خاصية text النص Press On Me ولما ادوس عليه on_press يعمل call للدالة press اللي بتعمل increment لمتغير اسمه i وتحط قيمة المتغير كـ text للزرار نفسه.



ولو عاوز تعرف اكتر عن الـ Button فكمل من هنا api-kivy.uix.button



Label

الـ label ده مكان مخصص عشان تعرض فيه نص فقط على عكس الزرار اللى بيستجيب لما تضغط عليه.

- عشان تعمله import

```
from kivy.uix.label import Label
```

- عشان تعمل object من الكلاس Label

```
label=Label(text="I'm A Label")
```

- الـ Label ليه نفس خصائص الـ button زى النص اللى فيها text ولون النص color وحجم الخط font_size وبتقدر تحدد الطول والعرض بالبيكسل بخاصية size أو بالنسبة المئوية بخاصية size_hint.

هناخد مثال على الـ Label وهو تعديل على السكريبت بتاع الـ Button.. هنحط فى الـ layout زرار Button و Label .. ولما تدوس على الزرار يعمل increment للمتغير i ويحط قيمته فى الـ Label.

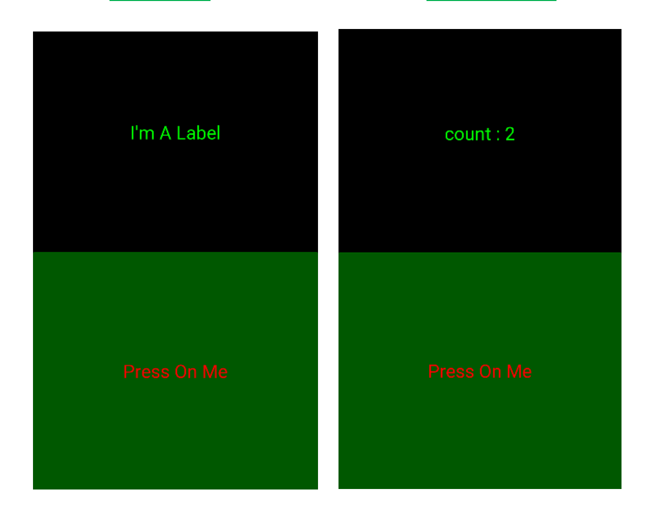
```
From kivy.app import App
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.boxlayout import BoxLayout

class myApp(App):
    i=0
```



```
    label=Label()
    def press(self,instance):
        self.label.text='count : '+str(self.i)
        self.i+=1
    def build(self):
        layout=BoxLayout(orientation='vertical')
        self.label=Label(
                text="I'm A Label",
            color=(0,1,0,1), #green
            font_size=70,
           )
        btn=Button(
                text="Press On Me",
            color=(1,0,0,1), #red
            background_color=(0,1,0,1), #green
            font_size=70,
            on_press=self.press,
           )
        layout.add_widget(self.label)
        layout.add_widget(btn)
        return layout
myApp().run()
```

Download script main.py , for android label.zip





TextInput

هو أداة إدخال أو مكان لكتابة النصوص من الكيبورد و يتم تسميته فى بعض اللغات والبرامجه التانيه EditText أو TextBox

- عشان تعمله import

```
from kivy.uix.textinput import TextInput
```

- عشان تعمل object من الكلاس TextInput.

```
label=Label(text='write anything')
```

multiline=True or False	بتحدد البوكس سطر واحد ولا متعدد الأسطر .
focus= True or False	لو True لما تفتح التطبيق هيكون المؤشر داخل البوكس ومستعد للكتاب.
allow_copy=True or False	السماح بتحديد نص من البوكس ونسخه
background_color=(R,G,B,Alpha)	لو الخلفية
color=(R,G,B,Alpha)	لون النص
text='my text'	النص بداخل البوكس
hint_text='text'	النص الإرشادى داخل البوكس (بيختفى لما تعمل focus)
hint_text_color=(R,G,B,Alpha)	لون النص الإرشادى



الأحداث events

- لما تكون عامل خاصية multiline=False وتدوس enter مش هينزل سطر جديد, لكن هيحصل ايفنت on_text_validate

```
def on_enter(instance):
    print('user pressed enter')

text = TextInput(text='Hello world', multiline=False)
text.bind(on_text_validate=on_enter)
```

- لما تكتب والنص داخل الـ TextInput يتغير بيحصل إيفنت اسمه text

```
def on_text(instance, value):
    print('typing..',instance.text)
text = TextInput()
text.bind(text=on_text)
```

ده تطبيق على الـ TextInput وعلى الـ event text..

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.uix.label import Label
from kivy.core.window import Window
Window.clearcolor = (1, 1, 1, 1)

players=['Essam El Hadary',
         'Wael Gomaa',
         'Ahmed Fathy',
         'Hossam Ghaly',
         'Emad Meteb',
         'Mohamed Zidan'
         ]
```



```
class TestApp(App):
    label=Label(color=(1,0,0,1),valign="top",size_hint=(1,0.90),font_size=70)
    def text_change(self,instance,value):
        self.label.text=""
        for player in players:
            if value in player.lower():
                self.label.text+=player+'\n'

    def build(self):
        layout=BoxLayout(orientation='vertical')
        txt=TextInput(hint_text='Search For Players',
                      multiline=False,
                      size_hint=(1,0.1),
                      color=(1,0,0,1),
                      font_size=80
                      )
        txt.bind(text=self.text_change)
        layout.add_widget(txt)
        layout.add_widget(self.label)
        return layout

TestApp().run()
```

Download script main.py , for android textinput.zip

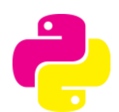
- فكرة التطبيق عبارة عن List اسمها players فيها اسامى اللاعيبة.. لما تكتب حرف أو جزء من اسم اللاعب فى الـ TextInput الحدث text ياخد النص ويبحث بيه فى الـ List ولو لقيه مطابق مع اسم لاعب او أكتر يحطه فى Label.



Search For Players	zi	emad
Essam El Hadary Wael Gomaa Ahmed Fathy Hossam Ghaly Emad Meteb Mohamed Zidan	Mohamed Zidan	Emad Meteb

موضوع الـ TextInput ده طويل وفيه حاجات كتير هتفيدك فلو عاوز تعرفها تقدر تكمل

في الوثيقة api-kivy.uix.textinput



الصـورة Image

عشـان تضيـف صـور فى التـطبيق بتسـتخـدم Image لـو صـورة محلـية علـى الجـهاز

ولـو صـورة من علـى الأنـترنت بتسـتخـدم AsyncImage

- عشـان تعملهـم import

```
from kivy.uix.image import Image
from kivy.uix.image import AsyncImage
```

- عشـان تعمل object من الـكلاس .

```
img=Image()
async_img= AsyncImage()
```

أهم الخصـائص

Source	اسـم أو مسـار الصـورة
allow_stretch = True or False	لـو True هيخـلى الصـورة تـملى علـى الـ Image Widget حتـى لو حجمهـا صغير
keep_ratio = True or False	لما يعمل stretch للصـورة هل يحافظ على معدل ثابت بين الطول والعـرض ولا لأ.
Nocache = True or False	لو ضـفت صـورة من الأنترنت بيعملها cashe في الابلكيشن عشان متتحملش كل مرة



```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button
from kivy.uix.togglebutton import ToggleButton
from kivy.uix.image import AsyncImage
from kivy.core.window import Window
Window.clearcolor = (1, 1, 1, 1)
src=['https://ma7moud3ly.github.io/1.jpg',
     'https://ma7moud3ly.github.io/2.jpg',
     'https://ma7moud3ly.github.io/3.jpg',
     'https://ma7moud3ly.github.io/4.jpg',
     'https://ma7moud3ly.github.io/5.jpg']

class TestApp(App):
    i=0
    img=AsyncImage(size_hint=(1,0.9),source=src[i],keep_ratio=False)
    img.source=src[2]
    def press(self,btn):
        if  btn.id=='1' and self.i<(len(src)-1):
            self.i+=1
            self.img.source=src[self.i]
        elif btn.id=='2' and self.i>0:
            self.i-=1
            self.img.source=src[self.i]
        elif btn.id=='3':self.img.allow_stretch=btn.state=='down'
        elif btn.id=='4':self.img.keep_ratio=btn.state=='down'

    def build(self):
        layout=BoxLayout(orientation='vertical')
        btns=BoxLayout(orientation='horizontal',size_hint=(1,0.1))
        btns.add_widget(Button(text='>>>',id='1',on_press=self.press))
        btns.add_widget(Button(text='<<<',id='2',on_press=self.press))

btns.add_widget(ToggleButton(text='Stretch',id='3',on_press=self.press))
        btns.add_widget(ToggleButton(text='Ratio',id='4',on_press=self.press))

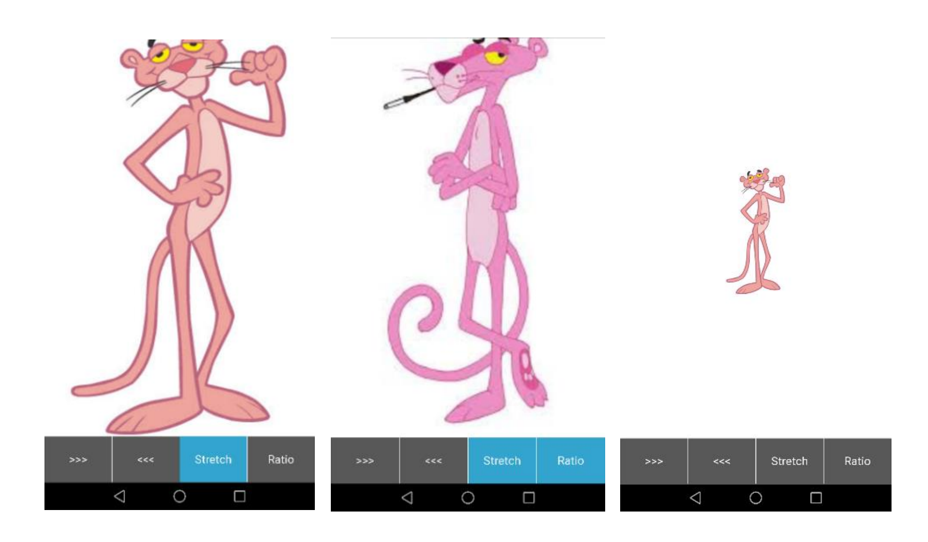
        layout.add_widget(self.img)
        layout.add_widget(btns)
        return layout

TestApp().run()
```

Download script main.py , for android image.zip



ده تطبيق عارض صور يعتمد على AsyncImage



ده كان مدخل بسيط لبرمجة تطبيقات الديسكتوب والموبايل عن طريقة لغة البايثون ومنصة Kivy .. لو كان فيه جزء تانى من الكتاب فأكيد هتناول مكتبة kivy بشكل مكثف اكتر من كده .. لكنك تقدر تكمل عن طريق الموقع الرسمى والوثيقة اللى هتلاقى فيها كمية كبيرة من الأمثلة و التطبيقات فى المعرض gallery وتقدر تشوف باقى الأدوات والـ widgets والـ api-kivy.uix والـ tutorial الجميلة دى kivy-crash-course.

الــمراجع

https://docs.python.org/3/

http://tutorialspoint.com/python

https://www.w3schools.com/python

تــــم بــحـــمــــد اللّه

PINK PYTHON 2018

break